João Alves Silva Júnior

First Steps in Homotopy Type Theory

Brasil

Fevereiro de 2014

João Alves Silva Júnior

First Steps in Homotopy Type Theory

Dissertação submetida ao Corpo Docente do Programa de Pós-Graduação do Departamento de Matemática da Universidade Federal de Pernambuco como parte dos requisitos para obtenção do grau de Mestre em Matemática.

Universidade Federal de Pernambuco

Centro de Ciências Exatas e da Natureza

Departamento de Matemática

Orientador: Ruy José Guerra Barreto de Queiroz

Brasil

Fevereiro de 2014

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

Silva Júnior, João Alves
First steps in homotopy type theory / João Alves Silva Júnior. - Recife: O Autor, 2014.
58 f., fig.

Orientador: Ruy José Guerra Barretto de Queiroz.
Dissertação (mestrado) - Universidade Federal de Pernambuco. CCEN, Matemática, 2014.

Inclui referências.

1. Matemática. 2. Álgebra. 3. Lógica matemática. I. Queiroz, Ruy José Guerra Barretto de (orientador). II. Título.

510 CDD (23. ed.) MEI2014 – 042

Dissertação submetida ao Corpo Docente do Programa de Pós-graduação do Departamento de Matemática da Universidade Federal de Pernambuco como parte dos requisitos necessários para a obtenção do Grau de Mestrado em Matemática.

Aprovado: _____

Ruy José Guerra Barreto de Queiroz*, UFPE*
Orientador

_____

Manoel José Machado Soares Lemos*, UFPE*

_____

Wilson Rosa de Oliveira Junior, UFRPE

FIRST STEPS IN HOMOTOPY TYPE THEORY

Por

João Alves Silva Junior

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA
DEPARTAMENTO DE MATEMÁTICA
Cidade Universitária – Tels. (081) 2126 - 8414 – Fax: (081) 2126 - 8410
RECIFE – BRASIL
27 de Fevereiro – 2014

To my parents

Acknowledgements

Above all, I am grateful to my family, especially my parents, João Alves (aka Dão) and Rosilene, for the steadfast support during all my student's life.

I would also like to express my gratitude to my adviser Ruy de Queiroz, for introducing me to this beautiful area of research. Thanks to Manoel Lemos, Wilson Rosa, and Peter Johnson for the examination of this dissertation and the given commentaries about it. Not less valuable was the knowledge and the maturity I acquired in the master's program, with the professors Tony Sousa, Seyed Hamid, and Fernando Xavier.

I warmly thank my colleagues from DMat/UFPE for the camaraderie and the exchange of ideas. I am indebted to Jaime for watching my pre-presentation; it was fundamental in my preparation.

Finally, thanks to Michael Shulman, Thorsten Altenkirch, Vladimir Voevodsky, among others, for some questions answered via internet.

Recife, Pernambuco
March 2014
João A. Silva Jr.

"Eventually I became convinced that the most interesting and important directions in current mathematics are the ones related to the transition into a new era which will be characterized by the widespread use of automated tools for proof construction and verification."

– Vladimir Voevodsky (VOEVODSKY, 2010)

Resumo

Em abril de 2013, o Programa de Fundamentos Univalentes do IAS, Princeton, lançou o primeiro livro em teoria homotópica de tipos, apresentando várias provas de resultados da teoria da homotopia em "um novo estilo de 'teoria de tipos informal' que pode ser lida e entendida por um ser humano, como um complemento à prova formal que pode ser checada por uma máquina". O objetivo desta dissertação é dar uma abordagem mais detalhada e acessível a algumas dessas provas. Escolhemos como leitmotiv uma versão tipo-teórica (originalmente proposta por Michael Shulman) de uma prova padrão de $\pi_1(\mathbb{S}^1) = \mathbb{Z}$ usando espaços de recobrimento. Um ponto crucial dela é o uso do "lema do achatamento" (*flattening lemma*), primeiramente formulado em generalidade por Guillaume Brunerie, cujo enunciado é bem complicado e cuja a prova é difícil, muito técnica e extensa. Enunciamos e provamos um caso particular desse lema, restringindo-o à mínima generalidade exigida pela demonstração de $\pi_1(\mathbb{S}^1) = \mathbb{Z}$. Também simplificamos outros resultados auxiliares, adicionamos detalhes a algumas provas e incluímos algumas provas originais de lemas simples como "composição de mapas preserva homotopia", "contrabilidade é uma invariante homotópica", "todo mapa entre tipos contráteis é uma equivalência", etc.

Palavras-chaves: teoria homotópica de tipos. fundamentos univalentes. grupo fundamental do círculo. lema do achatamento. cobertura universal do círculo.

Abstract

In April 2013, the Univalent Foundations Program, IAS, Princeton, released the first book on homotopy type theory, presenting several proofs of results from homotopy theory in "a new style of 'informal type theory' that can be read and understood by human beings, as a complement to a formal proof that can be checked by a machine." The objective of this dissertation is to give a more detailed and accessible approach to some of these proofs. We have chosen as leitmotif a type-theoretic version (originally proposed by Michael Shulman) of a standard proof of $\pi_1(\mathbb{S}^1) = \mathbb{Z}$ using covering spaces. A crucial point of it is the use of the flattening lemma, firstly formulated in generality by Guillaume Brunerie, whose statement is very complicated and whose proof is difficult, very technical and extensive. We state and prove a particular case of this lemma, restricting it to the minimum generality required by the proof of $\pi_1(\mathbb{S}^1) = \mathbb{Z}$. We also simplify other auxiliary results, add missing details to some proofs, and include some original proofs of simple lemmas such as "composition of maps preserves homotopy," "contractibility is a homotopy invariant," "every map between contractible types is an equivalence," etc.

Keywords: homotopy type theory. univalent foundations. fundamental group of the circle. flattening lemma. universal cover of the circle.

List of Figures

Contents

0 Introduction

Homotopy type theory is a recent field of research based on discovered connections between abstract homotopy theory and the branch of type theory from logic and theoretical computer science. As observed by Hofmann and Streicher (HOFMANN; STREICHER, 1998), in Martin-Löf intentional type theory (MARTIN-LÖF, 1975; MARTIN-LÖF, 1984; MARTIN-LÖF, 1998), each type A, endowed with its identity types $\mathsf{Id}_A(a, b)$, possesses a non-trivial structure, similar to a groupoid; in fact, a weak ∞-groupoid. Essentially the same is observed in a topological space when we consider its paths, path homotopies, homotopies between path homotopies, etc., all up to homotopy. This inspired Voevodsky (VOEVODSKY, 2006) and (independently) Awodey and Warren (AWODEY; WARREN, 2009) to develop the first works on homotopy type theory.

Voevodsky has constructed a model of type theory, using simplicial sets, that satisfies a property later called the *univalence axiom*, because it is now integrated to homotopy type theory as an axiom. In homotopy type theory, the univalence axiom says that isomorphic structures may be identified. So, it formalizes a common mathematical practice. The consequences of this axiom go far. Voevodsky has advocated the so called Univalent Foundations Program, suggesting new foundations for mathematics on the basis of homotopy type theory with the univalence axiom.

According (AWODEY, 2012), "the computational implementation of type theory allows computer verified proofs in homotopy theory", whereas "homotopy can be used as a tool to construct models of systems of logic". Computer scientists and mathematicians working in various areas are very excited about all the benefits that these connections promise. As said in (Univalent Foundations Program, 2013),

> "One can imagine a not-too-distant future when it will be possible for mathematicians to verify the correctness of their own papers by working within the system of univalent foundations, formalized in a proof assistant, and that doing so will become as natural as typesetting their own papers in TEX. In principle, this could be equally true for any other foundational system, but we believe it to be more practically attainable using univalent foundations (...)".

In this dissertation, we present some proofs in homotopy type theory which may serve as an introduction to the subject. Much of the text is heavily based on (Univalent Foundations Program, 2013).

As prerequisites, we assume some familiarity with topology, the sections 1.1–1.6, 1.11, and 1.12 of (Univalent Foundations Program, 2013), and some parts of (Univalent

Foundations Program, 2013, Chapter 6) as indicated in our Chapter 5. For reference, we summarize in the next section some things about informal type theory. Since we are primarily interested in Martin-Löf intentional type theory, we sometimes refer to this particular formulation simply as type theory.

0.1 Summary on informal type theory

Martin-Löf type theory is an extension of the typed λ-calculus. For an introduction to λ-calculus, we recommend (HINDLEY; SELDIN, 2008).

The basic judgements of type theory are $a : A$ (a has the type A) and $a \equiv b : A$ (a and b are equal terms of the type A). Here, equality between terms means *judgemental equality*, which is the same as *definitional equality*. This is a syntactic notion (see the last paragraph before section A.1.1 of (Univalent Foundations Program, 2013)). We usually write $a \equiv b : A$ as $a \equiv b$. Expressions of the form $A :\equiv B$ are used to define A as being B.

A *universe* is a type whose elements are types. We call the elements of a given universe its *small types*. Each term is associated to a unique small type, i.e., a type which is not a universe. We denote universes by $\mathcal{U}_0, \mathcal{U}_1, \mathcal{U}_2, \ldots$. Since there is a cumulative chain of universes, in any discussion inside type theory, we may suppose the existence of a universe $\mathcal{U}$ where all the other types are in.

The expression $B[u/x]$ indicates the term B' obtained by replacing all the free occurrences of the variable x in B by the term u, changing bound variables to avoid clashes.

0.1.1 Function types, or $\prod$-types

In the rules below, B and b may depend on x.

$\diamondsuit$ Formation rule: If $B : \mathcal{U}$ for any $x : A$, then $\prod_{(x:A)} B : \mathcal{U}$.

$\diamondsuit$ Introduction rule: If $b : B$ for any $x : A$, then $\lambda(x : A).b : \prod_{(x:A)} B$.

$\diamondsuit$ Elimination rule: For all $f : \prod_{(x:A)} B$ and all $a : A$, $f(a) : B[a/x]$.

$\diamondsuit$ Computation rule: For all $f : \prod_{(x:A)} B$ and all $a : A$, $(\lambda(x : A).f(x))(a) \equiv f(a)$.

$\diamondsuit$ Uniqueness rule: If $f : \prod_{(x:A)} B$, then $\lambda(x : A).f(x) \equiv f$.

The elements of $\prod_{(x:A)} B$ are called (dependent) *functions* or *maps*. When B does not depend on x, we denote $\prod_{(x:A)} B$ by $A \to B$. The elements of $A \to B$ are called *non-dependent functions*.

Given any type A, the function $\mathsf{id}_A :\equiv \lambda(x : A).x$ is called the *identity function* on A. Given functions $f : A \to B$ and $g : \prod_{(y:B)} C(y)$, with $C : B \to \mathcal{U}$, we define the *composite function* of f and g by $g \circ f :\equiv \lambda(x : A).g(f(x))$. Verify that composition of functions is associative, i.e., $f \circ (g \circ h) \equiv (f \circ g) \circ h$ whenever these instances of composition are well-defined.

A *type family* is a function whose values are types. In other words, a type family is an element of a type $A \to \mathcal{U}$, for some $A : \mathcal{U}$.

The notation $\lambda(x : A).f(x)$ is sometimes abbreviated to $\lambda x.f(x)$. Observe the other notational conventions adopted in (Univalent Foundations Program, 2013, Chapter 1), like

$$\begin{aligned} f(x, y) &:\equiv f(x)(y), \\ \prod_{x,y:A} B(x, y) &:\equiv \prod_{(x:A)} \prod_{(y:A)} B(x, y), \\ \prod_{x:A} B(x) \to C(x) &:\equiv \prod_{x:A} (B(x) \to C(x)), \\ A \to B \to C &:\equiv A \to (B \to C). \end{aligned}$$

0.1.2 Pair types, or $\sum$-types

♢ Formation rule: If $A : \mathcal{U}$ and $B : A \to \mathcal{U}$, then $\sum_{(x:A)} B(x) : \mathcal{U}$.

♢ Introduction rule: For all $\sum$-type $S :\equiv \sum_{(x:A)} B(x)$, with $A : \mathcal{U}$ and $B : A \to \mathcal{U}$, there is a function

$$\mathsf{intr}_S : \prod_{(x:A)} \prod_{(y:B(x))} S.$$

♢ Elimination rule: Given $A : \mathcal{U}$ and $B : A \to \mathcal{U}$, let S denote $\sum_{(x:A)} B(x)$. There is a map ind_S that gives a function

$$\mathsf{ind}_S(C, g) : \prod_{s:S} C(s)$$

for each

$$C : S \to \mathcal{U} \quad \text{and} \quad g : \prod_{(x:A)} \prod_{(y:B(x))} C((x, y)).$$

♢ Computation rule: In the context of the elimination rule above, if $x : A$ and $y : B(x)$, then

$$\mathsf{ind}_S(C, g, (x, y)) \equiv g(x)(y).$$

We usually denote $\mathsf{intr}_S(x, y)$ by (x, y). The elements of $\sum_{(x:A)} B(x)$ are called (dependent) *pairs*. When $B' : A \to \mathcal{U}$ is a constant function $\lambda(x : A).B$ for some $B : \mathcal{U}$, we denote $\sum_{(x:A)} B'(x)$ by $A \times B$. Types of the form $A \times B$ are called *product types*. Given $A, B : \mathcal{U}$, the type $A \times B$ is said to be the *cartesian product* of A and B.

Observe the notational conventions adopted in (Univalent Foundations Program, 2013, Chapter 1), like

$$\sum_{x,y:A} B(x,y) :\equiv \sum_{x:A}\sum_{y:A} B(x,y),$$
$$\sum_{x:A} B(x) \to C(x) :\equiv \sum_{x:A} (B(x) \to C(x)).$$

The elimination rule above, together with its computation rule, is also known as the *induction principle*, or the *dependent eliminator*, for dependent pair types. As a particular case of it, we have the *recursion principle*, or the *non-dependent eliminator*, for dependent pair types, which says that for any type S of the form $\sum_{(x:A)} B(x)$, there is a map rec_S that gives a function

$$\mathsf{rec}_S(C,g) :\equiv \mathsf{ind}_S(\lambda(s:S).C, g) : S \to C$$

for each

$$C : \mathcal{U} \quad \text{and} \quad g : \prod_{(x:A)} \prod_{(y:B(x))} C,$$

in such a way that

$$\mathsf{rec}_S(C, g, (x,y)) \equiv g(x)(y),$$

for any $x : A$ and $y : B$.

Lemma 0.1. *Associated to any type family $B : A \to \mathcal{U}$, there are functions*

$$\mathsf{pr}_1 : \left(\sum_{x:A} B(x)\right) \to A, \qquad \mathsf{pr}_2 : \prod_{w:\sum_{(x:A)} B(x)} B(\mathsf{pr}_1(w))$$

such that $\mathsf{pr}_1((x,y)) \equiv x$ and $\mathsf{pr}_2((x,y)) \equiv y$, for all $x : A$ and $y : B(x)$.

Proof. See (Univalent Foundations Program, 2013, Section 1.6). □

0.1.3 Identity types

♢ Formation rule: Given $A : \mathcal{U}$, for any $x, y : A$, we have $\mathsf{Id}_A(x,y) : \mathcal{U}$.

♢ Introduction rule: For all $A : \mathcal{U}$ and $x : A$, we have a certain element refl_x of $\mathsf{Id}_A(x,x)$.

♢ Elimination rule: Given $A : \mathcal{U}$, there is a map $\mathsf{ind}_{=_A}$ that gives a function

$$\mathsf{ind}_{=_A}(C,g) : \prod_{(x,y:A)} \prod_{(p:\mathsf{Id}_A(x,y))} C(x,y,p),$$

for each

$$C : \prod_{(x,y:A)} \prod_{(p:\mathsf{Id}_A(x,y))} \mathcal{U} \quad \text{and} \quad g : \prod_{x:A} C(x,x,\mathsf{refl}_x).$$

♢ Computation rule: In the context of the elimination rule above,

$$\mathsf{ind}_{=_A}(C, c, x, x, \mathsf{refl}_x) \equiv g(x).$$

We often denote $\mathsf{Id}_A(x, y)$ by $x =_A y$ or simply $x = y$. The elements of $\mathsf{Id}_A(x, y)$ are called *equalities*. Equalities of the form refl_x are called *reflexivities*.

The elimination rule above, together with its computation rule, is also known as the *path induction principle*. As a consequence of it, we have the *based path induction principle*[1], which says that for any $A : \mathcal{U}$ and $a : A$, there is a certain map $\mathsf{ind}'_{=_A}(a)$ that gives a function

$$\mathsf{ind}'_{=_A}(a, C, c) : \prod_{(x:A)} \prod_{(p:a=_A x)} C(x, p)$$

for each

$$C : \prod_{(x:A)} \prod_{(p:a=_A x)} \mathcal{U} \quad \text{and} \quad c : C(a, \mathsf{refl}_a)$$

in such a way that

$$\mathsf{ind}'_{=_A}(a, C, c, a, \mathsf{refl}_a) \equiv c.$$

Lemma 0.2 (Propositional uniqueness rule for $\sum$-types)**.** *For any $A : \mathcal{U}$ and $B : A \to \mathcal{U}$, there is a certain function*

$$\mathsf{uppt} : \prod_{w:\sum_{(x:A)} B(x)} (\mathsf{pr}_1(w), \mathsf{pr}_2(w)) = w$$

such that, for all $x : A$ and $y : B(x)$, $\mathsf{uppt}((x, y)) \equiv \mathsf{refl}_{(x,y)}$.

Proof. Let S denote $\sum_{(x:A)} B(x)$. Consider

$$g :\equiv \lambda(x : A).\lambda(y : B(x)).\mathsf{refl}_{(x,y)} : \prod_{(x:A)} \prod_{(y:B(x))} C((x, y)),$$

where $C :\equiv \lambda(w : S).\, ((\mathsf{pr}_1(w), \mathsf{pr}_2(w)) = w) : S \to \mathcal{U}$. Define uppt as being $\mathsf{ind}_S(C, g)$. □

This is a generalization of the function uppt presented in (Univalent Foundations Program, 2013, Section 1.5). It would be more correct to denote uppt by uppt_S, where S is the involved $\sum$-type. The same is true for the projection functions pr_1 and pr_2. But we usually ignore these details for simplicity.

[1] The based path induction principle is also a sufficient condition for the path induction principle, but for our purposes it suffices to observe the necessity.

0.1.4 Propositions as types

There is a correspondence between types and propositions according to which the expressions $\prod_{(x:A)}$ and $\sum_{(x:A)}$ are interpreted as $\forall x \in A$ and $\exists x \in A$, respectively. In particular, $A \times B$ and $A \to B$ are interpreted as "A and B" and "A implies B," respectively. Elements of a type are considered proofs of the correspondent proposition. The negation of a type A is defined as the type $A \to 0$, where 0 is the *empty type* (a type that has no elements). A type, viewed as a proposition, is true if it is inhabited and false if its negation is inhabited. Given types A and B, we say that A is a sufficient (repectively, necessary) condition for B if the type $A \to B$ (respectively $B \to A$) is inhabited. Two types A and B are said to be logically equivalent if both $A \to B$ and $B \to A$ are inhabited.

Since our type theory is constructive, so is its logical interpretation. For instance, our existential quantifier carry more information than the usual one, because we are not allowed to claim that something exists without exhibiting it. Hence, a translation of the axiom of choice into our type theory is automatically true. On the other hand, the law of double negation and law of excluded middle are not valid here.

1 Paths and Homotopies

As the reader probably know, in a topological space X, a path (from $x_0 \in X$ to $x_1 \in X$) is a continuous function $\gamma : [0,1] \to X$ (such that $\gamma(0) = x_0$ and $\gamma(1) = x_1$). The image of a path γ in X, called the *trace* of γ, is usually thought as the trajectory of a particle with position $\gamma(t)$ at the instant t. Sometimes we identify the path with its trace, so that we may assign geometric/physical properties/concepts to paths without extra definitions. Given paths γ and δ in X with $\gamma(1) = \delta(0)$,

- the inverse path (or simply the inverse) of γ is the path γ^{-1} given by $\gamma^{-1}(t) = \gamma(1-t)$;
- the concatenation (or the composition) of γ and δ is the path $\gamma \centerdot \delta$ given by

$$(\gamma \centerdot \delta)(t) = \begin{cases} \gamma(2t), & \text{if } 0 \leq t \leq 1/2; \\ \delta(2t-1), & \text{if } 1/2 < t \leq 1. \end{cases}$$

Note that γ^{-1} is a path from $\gamma(1)$ to $\gamma(0)$ with the same trace as γ. And $\gamma \centerdot \delta$ is something like γ followed by δ, each with double velocity.

Let γ and δ be paths in X, with $\gamma(0) = \delta(0)$ and $\gamma(1) = \delta(1)$. A (path) homotopy from γ to δ is a continuous map $H : [0,1] \times [0,1] \to X$ such that $H(s,0) = \gamma(s)$, $H(s,1) = \delta(s)$, and $H(k,t) = \gamma(k) = \delta(k)$,for all $k \in 0,1$ and $t \in [0,1]$. If there is such a H, we write $H : \gamma \sim \delta$, or simply $\gamma \sim \delta$, and we say that γ is homotopic to δ (via H). So, we have defined a binary relation $\sim$ on set of all paths in X from x_1 to x_2, for any $x_1, x_2 \in X$. This is an equivalence relation and the operations of inversion and concatenation of paths are well-defined on the equivalence classes. That is, we may concatenate and invert equivalence classes by operating with (any of) its representatives. More precisely, if $\gamma \sim \gamma'$, $\delta \sim \delta'$ and $[\alpha]$ denotes the equivalence class of a given path α with respect to $\sim$, then $[\gamma^{-1}] = [\gamma'^{-1}]$ and $[\gamma \centerdot \delta] = [\gamma' \centerdot \delta']$, so that we may define $[\gamma]^{-1} = [\gamma^{-1}]$ and $[\gamma] \centerdot [\delta] = [\gamma \centerdot \delta]$.

In this chapter, we interpret equalities as paths up to homotopy, i.e., as equivalence classes of the path homotopy relation. Due to this interpretation, elements of identity types are often called *paths*. Reflexivities are interpreted as (and therefore called) *constant paths*.

1.1 Path operations

Lemma 1.1. *For any type A, there are functions*

(a) $\mathsf{pinv}_A : \prod_{(x,y:A)}(x = y) \to (y = x)$,

(b) $\mathsf{conc}_A : \prod_{(x,y,z:A)}(x = y) \to (y = z) \to (x = z)$,

called path inversion *and* path concatenation *(or* path composition*), respectively, such that* $\mathsf{pinv}_A(x, x, \mathsf{refl}_x) \equiv \mathsf{refl}_x$ *and* $\mathsf{conc}_A(x, x, x, \mathsf{refl}_x, \mathsf{refl}_x) \equiv \mathsf{refl}_x$*, for all* $x : A$.

Proof.

(a) We must construct an element pinv_A of the type $\prod_{(x,y:A)}(x = y) \to (y = x)$, which is definitionally equal to $\prod_{(x,y:A)} \prod_{(p:x=y)} C(x, y, p)$, for $C : \prod_{(x,y:A)} \prod_{(p:x=y)} \mathcal{U}$ given by $C(x, y, p) :\equiv (y = x)$. The definition of pinv_A must satisfy the computation rule $\mathsf{refl}_x \equiv \mathsf{pinv}_A(x, x, \mathsf{refl}_x)$. By path induction, it suffices to exhibit a $g : \prod_{(x:A)} C(x, x, \mathsf{refl}_x)$ satisfying $g(x) \equiv \mathsf{refl}_x$, so that we can define $\mathsf{pinv}_A :\equiv \mathsf{ind}_{=_A}(C, g)$. These conditions are satisfied by $g :\equiv \lambda(x : A).\mathsf{refl}_x$. So, the proof is concluded.

(b) Let $C_1 : \prod_{(x,y:A)} \prod_{(p:x=y)} \mathcal{U}$ be given by $C_1(x, y, p) :\equiv \prod_{(z:A)} \prod_{(q:y=z)} x = z$. Note that the type $T :\equiv \prod_{(x,y,z:A)}(x = y) \to (y = z) \to (z = x)$, as a proposition, is a necessary condition for $S :\equiv \prod_{(x,y:A)} \prod_{(p:x=y)} C_1(x, y, p)$. In fact, it is easy to see that these are logically equivalent, but for our purposes here it is sufficient to observe that we have a function $\psi : S \to T$, which is defined by $\psi(g, x, y, z, p) :\equiv g(x, y, p, z)$, for any $g : S$, $x, y, z : A$, and $p : x = y$. To prove S by path induction, it suffices to find an element g_1 of the type $\prod_{(x:A)} C_1(x, x, \mathsf{refl}_x)$, which is definitionally equal to $R :\equiv \prod_{(x,z:A)} \prod_{(q:x=z)} C_2(x, z, q)$, for $C_2 : \prod_{(x,z:A)} \prod_{(q:x=z)} \mathcal{U}$ given by $C_2(x, z, q) :\equiv (x = z)$. To prove R by path induction, it suffices to construct a $g_2 : \prod_{(x:A)} C_2(x, x, \mathsf{refl}_x) \equiv \prod_{(x:A)}(x = x)$. The obvious choice is $g_2 :\equiv \lambda(x : A).\mathsf{refl}_x$. Now, we may define $g_1 :\equiv \mathsf{ind}_{=_A}(C_2, g_2) : R$, $g_0 :\equiv \mathsf{ind}_{=_A}(C_1, g_1) : S$ and $\mathsf{conc}_A :\equiv \psi(g_0) : T$. So, for all $x : A$,

$$\mathsf{conc}_A(x, x, x, \mathsf{refl}_x, \mathsf{refl}_x) \equiv g_0(x, x, \mathsf{refl}_x, x, \mathsf{refl}_x) \equiv g_1(x, x, \mathsf{refl}_x) \equiv g_2(x) \equiv \mathsf{refl}_x. \quad \square$$

We usually denote $\mathsf{conc}_A(x, y, z, p, q)$ by $p \centerdot q$ and $\mathsf{pinv}_A(x, y, p)$ by p^{-1}, leaving the arguments $x, y, z : A$ omitted. We call $p \centerdot q$ the *concatenation* or the *composite* of p and q. The path p^{-1} is called the *inverse* of p. In the rest of this dissertation, we do more endpoints omissions like these, without explicit warning. For example, the function ap_f of Lemma 1.3 is actually an element of $\prod_{(x,y:A)} \prod_{(p:x=y)} f(x) = f(y)$ and the notation $\mathsf{ap}_f(p)$ is an abbreviation for $\mathsf{ap}_f(x, y, p)$.

The proof by path induction given for Lemma 1.1 is too formal and it may be tedious to prove more complicated claims in that way. More practical styles of proof by path induction are used in the next lemmas.

Lemma 1.2. *For all* $p : x =_A y$*,* $q : y =_A z$*, and* $r : z =_A w$*,*

(a) $p = p \centerdot \mathsf{refl}_y$;

(b) $p = \mathsf{refl}_x \centerdot p$;

(c) $p^{-1} \centerdot p = \mathsf{refl}_y$;

(d) $p \centerdot p^{-1} = \mathsf{refl}_x$;

(e) $(p^{-1})^{-1} = p$;

(f) $p \centerdot (q \centerdot r) = (p \centerdot q) \centerdot r$.

Proof. By the principle of path induction, it suffices to consider the case when $x \equiv y \equiv z$ and $p \equiv q \equiv r \equiv \mathsf{refl}_x$. But, in this case, all the equalities above hold trivially, with both sides being judgmentally equal, by the computation rules of concatenation and inversion of paths.

For example, if $x \equiv y$, then $\mathsf{refl}_x \equiv \mathsf{refl}_y$. Since $\mathsf{refl}_x \equiv \mathsf{refl}_x \centerdot \mathsf{refl}_x$, it follows that $\mathsf{refl}_x \equiv \mathsf{refl}_x \centerdot \mathsf{refl}_y$, so that $(\mathsf{refl}_x = \mathsf{refl}_x) \equiv (\mathsf{refl}_x = \mathsf{refl}_x \centerdot \mathsf{refl}_y)$. Now, by $\mathsf{refl}_{\mathsf{refl}_x} : \mathsf{refl}_x = \mathsf{refl}_x$, we deduce $\mathsf{refl}_{\mathsf{refl}_x} : \mathsf{refl}_x = \mathsf{refl}_x \centerdot \mathsf{refl}_y$. This proves item (a), since we may suppose without loss of generality that $p \equiv \mathsf{refl}_x$ (this is the induction hypothesis). More formally,

$$\mathsf{ind}_{=_A}(C, g, x, y, p) : p = p \centerdot \mathsf{refl}_y,$$

where $g : \prod_{(x:A)} C(x, x, \mathsf{refl}_x)$ and $C : \prod_{(x,y:A)} \prod_{(p:x=y)} \mathcal{U}$ are defined by $g(x) :\equiv \mathsf{refl}_{\mathsf{refl}_x}$ and $C(x, y, p) :\equiv (p = p \centerdot \mathsf{refl}_y)$.

We outline the proof of one more item and leave the other as exercises to the reader. Since $\mathsf{refl}_x \centerdot \mathsf{refl}_x \equiv \mathsf{refl}_x$, we have that $\mathsf{refl}_x \centerdot (\mathsf{refl}_x \centerdot \mathsf{refl}_x) \equiv \mathsf{refl}_x \centerdot \mathsf{refl}_x \equiv (\mathsf{refl}_x \centerdot \mathsf{refl}_x) \centerdot \mathsf{refl}_x$. Therefore,

$$\mathsf{refl}_{\mathsf{refl}_x \centerdot (\mathsf{refl}_x \centerdot \mathsf{refl}_x)} : \mathsf{refl}_x \centerdot (\mathsf{refl}_x \centerdot \mathsf{refl}_x) = (\mathsf{refl}_x \centerdot \mathsf{refl}_x) \centerdot \mathsf{refl}_x.$$

From this, by path induction, one obtains a proof for $p \centerdot (q \centerdot r) = (p \centerdot q) \centerdot r$. □

1.2 Action of a function on a path

Lemma 1.3. *For any $f : A \to B$ and $p : x =_A y$, there is a path $\mathsf{ap}_f(p) : f(x) =_B f(y)$. The function ap_f defined in this way is such that $\mathsf{ap}_f(\mathsf{refl}_x) \equiv \mathsf{refl}_{f(x)}$ for all $x : A$.*

Proof. Let $p : x =_A y$ be arbitrary. By path induction, we may assume without loss of generality that $x \equiv y$ and $p \equiv \mathsf{refl}_x$. Since $x \equiv y$, we have that $f(x) \equiv f(y)$. Then we may define $\mathsf{ap}_f(p) :\equiv \mathsf{refl}_{f(x)}$. Since $p \equiv \mathsf{refl}_x$, the computation rule $\mathsf{ap}_f(\mathsf{refl}_x) \equiv \mathsf{refl}_{f(x)}$ is satisfied. □

In the homotopical interpretation of type theory, functions correspond to continuous maps and $\mathsf{ap}_f(p)$ is the image of the path p under the map f.

Lemma 1.4. *Given $f : A \to B$, $g : B \to C$, $p : x =_A y$, $q : y =_A z$, and $r : z =_A w$,*

(a) $\mathsf{ap}_f(p \cdot q) = \mathsf{ap}_f(p) \cdot \mathsf{ap}_f(q)$;

(b) $\mathsf{ap}_f(p^{-1}) = \mathsf{ap}_f(p)^{-1}$;

(c) $\mathsf{ap}_{g \circ f}(p) = \mathsf{ap}_g(\mathsf{ap}_f(p))$;

(d) $\mathsf{ap}_{\mathsf{id}_A}(p) = p$.

Proof. Apply path induction on p and q, using the computation rules of the involved functions. □

We want to generalize Lemma 1.3 for dependent functions, but in this case $f(x)$ and $f(y)$ may have different types, and we still have no definition for paths between points of different types. An idea for this generalization is to consider a path between points $u : P(x)$ and $v : P(y)$ as a path between the elements (x, u) and (y, v) of $\sum_{(a:A)} P(a)$. In Theorem 3.5, we give another characterization for this notion of paths in a type family.

Lemma 1.5. *To each $f : \prod_{(a:A)} B(a)$ and each $p : x =_A y$, we can assign a path*

$$\mathsf{apd}'_f(p) : (x, f(x)) =_{\sum_{(a:A)} B(a)} (y, f(y))$$

in such a way that the function apd'_f so defined satisfies $\mathsf{apd}'_f(\mathsf{refl}_x) \equiv \mathsf{refl}_{(x,f(x))}$ for all $x : A$.

Proof. Just apply Lemma 1.3 for the function $f' : A \to \sum_{(a:A)} B(a)$ defined by $f'(x) :\equiv (x, f(x))$. □

1.3 Generic transport lemmas

In this section, we define the operation of transport along a path. We state some lemmas about transport which will be often used in the rest of this dissertation. The proofs of these lemmas are simple path inductions, which we have omitted or just outlined. Other transport lemmas are given in Section 3.1.

Lemma 1.6. *For any type family $P : A \to \mathcal{U}$, there is a map transp^P that assigns a function $\mathsf{transp}^P(p) : P(x) \to P(y)$ to each $p : x =_A y$. Moreover, $\mathsf{transp}^P(\mathsf{refl}_x) \equiv \mathsf{id}_{P(x)}$, for all $x : A$.*

Proof. In the case when $x \equiv y$, we have $P(x) \equiv P(y)$, so that we may define $\mathsf{transp}^P(\mathsf{refl}_x) :\equiv \mathsf{id}_{P(x)}$. The general assertion follows immediately by path induction. □

We sometimes use the notation p_* for $\mathsf{transp}^P(p)$, leaving the type family P implicit. Given $u : P(x)$, we call $\mathsf{transp}^P(p, u)$ the *transport* of u along p with respect to P.

From the logical viewpoint, Lemma 1.6 tells that any property of x holds for y provided that $x = y$. By symmetry of equality (Lemma 1.1(a)), it follows that any property of y also holds for x when $x = y$. So, we have that identical objects are logically indistinguishable.

Lemma 1.7. *Given $P : A \to \mathcal{U}$, $p : x =_A y$, and $q : y =_A z$, for any $u : P(x)$, we have a path*

$$\mathsf{ittransp}^P(p, q, u) : q_*(p_*(u)) = (p \centerdot q)_*(u).$$

Moreover, $\mathsf{ittransp}^P(\mathsf{refl}_x, \mathsf{refl}_x, u) \equiv \mathsf{refl}_u$, for any $x : A$ and $u : P(x)$.

Proof. An easy path induction on p and q. □

Corollary 1.8. *For all $P : A \to \mathcal{U}$, $p : x =_A y$, $u : P(x)$, and $v : P(y)$, we have paths*

(a) $\mathsf{ittranspinv}_1^P(p, u) : {p^{-1}}_*(p_*(u)) = u$;

(b) $\mathsf{ittranspinv}_2^P(p, v) : p_*({p^{-1}}_*(v)) = v$.

Proof. By Lemmas 1.2(c) and 1.1(a), we have a path $r : p \centerdot p^{-1} = \mathsf{refl}_x$. Consider $e \equiv \lambda(s : x = x).s_*(u)$ and observe that $\mathsf{ap}_e(r) : (p \centerdot p^{-1})_*(u) = (\mathsf{refl}_x)_*(u) \equiv \mathsf{id}_{P(x)}(u) \equiv u$. So, we may define $\mathsf{ittranspinv}^P(p, u) :\equiv \mathsf{ittransp}_1^P(p, p^{-1}, u) \centerdot \mathsf{ap}_e(r)$. This proves part (a). The other part is proved analogously. □

Lemma 1.9. *Given $f : A \to B$, $P : A \to \mathcal{U}$, $p : x =_A y$, and $u : P(f(x))$,*

$$\mathsf{transp}^{P \circ f}(p, u) = \mathsf{transp}^P(\mathsf{ap}_f(p), u).$$

Proof. Immediate by path induction on p. □

1.4 Homotopies between maps

In set theory, two functions $f, g : A \to B$ are equal if and only if $f(x) = g(x)$ for all $x \in A$; but a translation of this (logical) equivalence into type theory is not automatically true (see Section 2.2). More precisely, under the type-theoretic rules so far introduced, we cannot define a function from $\prod_{(x:A)} f(x) = g(x)$ to $f = g$, although there is a map in the reverse direction. However, if the type $\prod_{(x:A)} f(x) = g(x)$ is inhabited, then f and g are very similar from the homotopical perspective.

As we have said in Section 1.2, in homotopy type theory, functions between types are considered continuous maps between topological spaces and proofs of equalities are viewed

as paths. From this viewpoint, it is natural to consider each element H of $\prod_{(x:A)} f(x) = g(x)$ as a family of paths γ_x from $f(x)$ to $g(x)$, with x varying on A. Equivalently, H is a homotopy between f and g, i.e., a continuous function $H : A \times [0,1] \to B$ such that $H(x,0) = f(x)$ and $H(x,1) = g(x)$, for all $x \in A$.

We generalize this notion of homotopy for dependent functions.

Definition 1.1. Let $A : \mathcal{U}$ and $B : A \to \mathcal{U}$. Given $f, g : \prod_{(x:A)} B(x)$, a homotopy from f to g (or between f and g) is a function $H : \prod_{(x:A)} f(x) = g(x)$. We denote

$$(f \sim g) :\equiv \prod_{x:A} f(x) = g(x).$$

The previous definition introduces a binary relation $\sim$ on each dependent function type $\prod_{(x:A)} B(x)$. This is an equivalence relation, as we prove now.

Lemma 1.10 (Homotopy between maps is an equivalence relation)**.** *Given* $f, g, h : \prod_{(x:A)} B(x)$,

(a) $f \sim f$;

(b) if $f \sim g$, *then* $g \sim f$;

(c) if $f \sim g$ *and* $g \sim h$, *then* $f \sim h$.

Proof. Just verify that

$$\lambda(x : A).\mathsf{refl}_{f(x)} : f \sim f$$
$$\lambda(H : f \sim g).\lambda(x : A).H(x)^{-1} : (f \sim g) \to (g \sim f)$$
$$\lambda(H_1 : f \sim g).\lambda(H_2 : g \sim h).\lambda(x : A).H_1(x) \centerdot H_2(x) : (f \sim g) \to (g \sim h) \to (f \sim h)$$ □

Lemma 1.11 (Composition of maps preserves homotopy)**.** *Given* $f_1, f_2 : A \to B$ *and* $g_1, g_2 : B \to C$, *if* $F : f_1 \sim f_2$ *and* $G : g_1 \sim g_2$, *then* $H : g_1 \circ f_1 \sim g_2 \circ f_1$, *where* H *is defined by* $H(x) \equiv \mathsf{ap}_{g_1}(F(x)) \centerdot G(f_2(x))$.

Proof. For all $x : A$, we have:

$$\begin{aligned} g_1(f_1(x)) &= g_1(f_2(x)) && \text{by } \mathsf{ap}_{g_1}(F(x)) \\ &= g_2(f_2(x)), && \text{by } G(f_2(x)) \end{aligned}$$

so that $\mathsf{ap}_{g_1}(F(x)) \centerdot G(f_2(x)) : g_1(f_1(x)) = g_2(f_2(x))$, i.e.,

$$H(x) : (g_1 \circ f_1)(x) = (g_2 \circ f_2)(x).$$ □

Lemma 1.12 (Homotopies are natural transformations)**.** *Given $f, g : A \to B$, let H be a homotopy from f to g. For all $p : x =_A y$, we have*

$$H(x) \centerdot \mathsf{ap}_g(p) = \mathsf{ap}_f(p) \centerdot H(y). \tag{1.1}$$

Proof. In the case when $x \equiv y$ and $p \equiv \mathsf{refl}_x$, the type (1.1) is definitionally equal to

$$H(x) \centerdot \mathsf{refl}_{g(x)} = \mathsf{refl}_{f(x)} \centerdot H(x), \tag{1.2}$$

which is a necessary condition for $H(x) = H(x)$. In fact, we can define a function

$$F : (H(x) = H(x)) \to \Big(H(x) \centerdot \mathsf{refl}_{g(x)} = \mathsf{refl}_{f(x)} \centerdot H(x)\Big)$$

by $F(p) :\equiv r \centerdot p \centerdot s$, where $r : H(x) \centerdot \mathsf{refl}_{g(x)} = H(x)$ and $s : \mathsf{refl}_{f(x)} \centerdot H(x) = H(x)$ are given by Lemma 1.2. So, since $\mathsf{refl}_{H(x)} : H(x) = H(x)$, we see that (1.2) has a proof. Therefore, by path induction, (1.1) is true for all $p : x =_A y$. □

Corollary 1.13. *If $f : A \to A$ and $H : f \sim \mathsf{id}_A$, then $H(f(x)) = \mathsf{ap}_f(H(x))$ for all $x : A$.*

Proof. Replacing (simultaneously) x by $f(x)$, y by x, p by $H(x)$, and g by id_A in (1.1), we have that $H(f(x)) \centerdot H(x) = \mathsf{ap}_f(H(x)) \centerdot H(x)$. By concatenating both sides of this equality with $H(x)^{-1}$ to the right, it follows that

$$(H(f(x)) \centerdot H(x)) \centerdot H(x)^{-1} = (\mathsf{ap}_f(H(x)) \centerdot H(x)) \centerdot H(x)^{-1},$$

by Lemma 1.3. Then, by Lemma 1.2,

$$\begin{aligned} H(f(x)) = H(f(x)) \centerdot \mathsf{refl}_{f(x)} &= H(f(x)) \centerdot (H(x) \centerdot H(x)^{-1}) \\ &= (H(f(x)) \centerdot H(x)) \centerdot H(x)^{-1} = (\mathsf{ap}_f(H(x)) \centerdot H(x)) \centerdot H(x)^{-1} \\ &= \mathsf{ap}_f(H(x)) \centerdot (H(x) \centerdot H(x)^{-1}) = \mathsf{ap}_f(H(x)) \centerdot \mathsf{refl}_{f(x)} = \mathsf{ap}_f(H(x)). \end{aligned}$$

□

2 Equivalences

Two topological spaces A and B are said to be homotopy equivalent if there are continuous maps $f : A \to B$ and $g : B \to A$ such that $f \circ g$ and $g \circ f$ are homotopic to the identity maps on B and A, respectively. In Section 2.1, we translate this concept into type theory and prove some lemmas about it. Next, in Section 2.2, we present the axioms of function extensionality and univalence.

2.1 Homotopy equivalences

Definition 2.1. A *quasi-inverse* of a function $f : A \to B$ is a function $g : B \to A$ together with homotopies $h : f \circ g \sim \mathsf{id}_B$ and $k : g \circ f \sim \mathsf{id}_A$. More precisely, the quasi-inverses of $f : A \to B$ are all the elements of the type

$$\mathsf{qinv}(f) :\equiv \sum_{g:B\to A} ((f \circ g \sim \mathsf{id}_B) \times (g \circ f \sim \mathsf{id}_A)).$$

Informally, we say that $g : B \to A$ is a quasi-inverse of $f : A \to B$ if $g \circ f \sim \mathsf{id}_A$ and $f \circ g \sim \mathsf{id}_B$.

Definition 2.2. A (homotopy) equivalence from A to B (or between A and B) is a pair (f, q), where $f : A \to B$ and $q : \mathsf{qinv}(f)$. We denote the type of equivalences from A to B by $A \simeq B$. Symbolically,

$$(A \simeq B) :\equiv \sum_{f:A\to B} \mathsf{qinv}(f).$$

Thus, if $\varepsilon : A \simeq B$, then $\mathsf{pr}_2(\varepsilon)$ is a quasi-inverse of $\mathsf{pr}_1(\varepsilon)$, so that $\mathsf{pr}_2(\varepsilon, \mathsf{pr}_1(\varepsilon, x)) = x$ and $\mathsf{pr}_1(\varepsilon, \mathsf{pr}_2(\varepsilon, y)) = y$, for all $x : A$ and $y : B$. Informally, we say that a function $f : A \to B$ is an equivalence if it has a quasi-inverse. But it is necessary to be careful to not confuse this informal convention with the official definition, according to which an equivalence is not a function, but a pair.

Example 2.1. *Path inversion is an equivalence. For any $x, y : A$, the function $\lambda(p : x = y).p^{-1}$ is a quasi-inverse of itself (by Lemma 1.2(e)).*

Example 2.2. *Concatenations to the right and to the left of a fixed path are equivalences. More precisely, given a path $p : x =_A y$ and points $x', y' : A$, the functions $\lambda(r : x' = x).r \centerdot p$ and $\lambda(r : x' = x).r^{-1} \centerdot p$ are quasi-inverses of each other, as well as $\lambda(s : y = y').p \centerdot s$ and $\lambda(s : y = y').p \centerdot s^{-1}$. The verification is immediate by Lemma 1.2.*

Example 2.3. *Given a type family $P : A \to \mathcal{U}$ and a path $p : x =_A y$, the function $\mathsf{transp}^P(p)$ is an equivalence with quasi-inverse $\mathsf{transp}^P(p^{-1})$. In fact, for all $u : P(x)$ and $v : P(y)$, we have $\mathsf{transp}^P(p^{-1}, \mathsf{transp}^P(p, u)) = u$ and $\mathsf{transp}^P(p, \mathsf{transp}^P(p^{-1}, v)) = v$, by Lemma 1.8.*

Definition 2.2 introduces a binary relation $\simeq$ on $\mathcal{U}$. This is an equivalence relation.

Lemma 2.1 (Homotopy equivalence is an equivalence relation)**.** *For all $A, B, C : \mathcal{U}$,*

(a) $A \simeq A$;

(b) if $A \simeq B$, then $B \simeq A$;

(c) if $A \simeq B$ and $B \simeq C$, then $A \simeq C$.

Proof.

(a) Just verify that $(\mathsf{id}_A, (\mathsf{id}_A, (\lambda a.\mathsf{refl}_a, \lambda a.\mathsf{refl}_a))) : A \simeq A$.

(b) Note that if $(f, (g, (h, k))) : A \simeq B$, then $(g, (f, (k, h))) : B \simeq A$. Moreover, by successive applications of Lemma 0.2, every element of $A \simeq B$ is equal to $(f, (g, (h, k)))$ for some (specific) $f : A \to B$, $g : B \to A$, $h : f \circ g \sim \mathsf{id}_B$ and $k : g \circ f \sim \mathsf{id}_A$. So, we can define a function $F : (A \simeq B) \to (B \simeq A)$ by $F(v) :\equiv (g, (f, (k, h)))$, where $(f, (g, (h, k))) = v$.

(c) Given $(f_1, (g_1, (h_1, k_1))) : A \simeq B$ and $(f_2, (g_2, (h_2, k_2))) : B \simeq C$, consider

$$q_1 :\equiv (\lambda z : C).\mathsf{ap}_{f_2}(h_1(g_2(z))) \centerdot h_2(z), \tag{2.1}$$

$$q_2 :\equiv (\lambda x : A).\mathsf{ap}_{g_1}(k_2(f_1(x))) \centerdot k_1(x), \tag{2.2}$$

and verify that $(f_2 \circ f_1, (g_1 \circ g_2, (q_1, q_2))) : A \simeq C$. Therefore, we can define a function $G : (A \simeq B) \to (B \simeq C) \to (A \simeq C)$ by $G(v, w) :\equiv (f_2 \circ f_1, (g_1 \circ g_2, (q_1, q_2)))$, where $(f_1, (g_1, (h_1, k_1)))$ and $(f_2, (g_2, (h_2, k_2)))$, given by Lemma 0.2, are equal to v and w, respectively. □

Definition 2.3. For any $v : A \simeq B$ and $w : B \simeq C$, the previous lemma (items (b) and (c), respectively) gives equivalences $F(v) : B \simeq A$ and $G(v, w) : A \simeq C$. We denote these by v^{-1} and $w \circ v$, respectively. We call v^{-1} the inverse (equivalence) of v and $w \circ v$ the composite (equivalence) of v and w.

Informally, one may consider elements of $A \simeq B$ as functions from A to B that have quasi-inverses, or denote by f^{-1} any function from B to A such that both $f \circ f^{-1}$ and $f \circ f^{-1}$ are homotopic to identity functions, for a given $f : A \to B$, but we avoid this kind of abuse, especially when writing symbolically.

Example 2.4. *Consider a path $p : x =_A y$ and a point $z : A$. By Examples 2.1 and 2.2, we have equivalences $\alpha : (x = z) \simeq (z = x)$ and $\beta : (z = x) \simeq (z = y)$ such that $\mathsf{pr}_1(\alpha, r) \equiv r^{-1}$ and $\mathsf{pr}_1(\beta, s) \equiv s \centerdot p$. Then $\beta \circ \alpha : (x = z) \simeq (z = y)$ satisfies $\mathsf{pr}_1(\beta \circ \alpha, r) \equiv r^{-1} \centerdot p$.*

In the rest of this section, we prove some simple lemmas about equivalences to be used in the next chapters.

Lemma 2.2. *Given a type A and paths $p : a =_A b$ and $q : c =_A d$, we have equivalences $\alpha : (a =_A c) \simeq (b =_A d)$ and $\beta : (a =_A d) \simeq (c =_A b)$ such that*

$$\mathsf{pr}_1(\alpha) \equiv \lambda(r : a = c).p^{-1} \centerdot r \centerdot q, \qquad \mathsf{pr}_2(\alpha) \equiv \lambda(s : b = d).p \centerdot s \centerdot q^{-1},$$
$$\mathsf{pr}_1(\beta) \equiv \lambda(r : a = d).q \centerdot r^{-1} \centerdot p, \qquad \mathsf{pr}_2(\beta) \equiv \lambda(s : c = b).p \centerdot s^{-1} \centerdot q.$$

In particular, if $a \equiv b$, $p \equiv \mathsf{refl}_a$, $d \equiv c$, and $q \equiv \mathsf{refl}_c$, then $\beta : (a =_A c) \simeq (c =_A a)$ and this is essentially Example 2.1. If either $p \equiv \mathsf{refl}_a$ or $q \equiv \mathsf{refl}_c$, then α is essentially Example 2.2.

Proof. It is immediate to verify that $\mathsf{pr}_1(\alpha) \circ \mathsf{pr}_2(\alpha)$, $\mathsf{pr}_2(\alpha) \circ \mathsf{pr}_1(\alpha)$, $\mathsf{pr}_1(\beta) \circ \mathsf{pr}_2(\beta)$, and $\mathsf{pr}_2(\beta) \circ \mathsf{pr}_1(\beta)$ are homotopic to identity functions, by Lemma 1.2. □

Lemma 2.3 ($\sum$-types are "associative")**.** *For any $C : \left(\sum_{(a:A)} B(a)\right) \to \mathcal{U}$, with $A : \mathcal{U}$ and $B : A \to \mathcal{U}$,*

$$\sum_{(a:A)} \sum_{(b:B(a))} C((a, b)) \simeq \sum_{q:\sum_{(a:A)} B(a)} C(q). \tag{2.3}$$

Proof. Let M and N denote the left and the right sides of (2.3), respectively. The idea of the proof is simply to define $f : M \to N$ and $g : N \to M$ so that

$$f((a, (b, c))) \equiv ((a, b), c), \qquad g(((a, b), c)) \equiv (a, (b, c)),$$

and hence

$$f(g(((a, b), c))) \equiv ((a, b), c), \qquad g(f((a, (b, c)))) \equiv (a, (b, c)),$$

for all $a : A$, $b : B(a)$, and $c : B((a, b))$. These conditions hold for

$$f :\equiv \lambda(v : M).((\mathsf{pr}_1 v, \mathsf{pr}_1 \mathsf{pr}_2 v), \mathsf{pr}_2 \mathsf{pr}_2 v),$$
$$g :\equiv \lambda(w : N).(\mathsf{pr}_1 \mathsf{pr}_1 w, (\mathsf{pr}_2 \mathsf{pr}_1 w, \mathsf{pr}_2 w)),$$

where, for brevity, we have omitted the parentheses around arguments of projection functions. Observe that

$$f(g(w)) \equiv ((\mathsf{pr}_1 \mathsf{pr}_1 w, \mathsf{pr}_2 \mathsf{pr}_1 w), \mathsf{pr}_2 w), \qquad g(f(v)) \equiv (\mathsf{pr}_1 v, (\mathsf{pr}_1 \mathsf{pr}_2 v, \mathsf{pr}_2 \mathsf{pr}_2 v)).$$

Moreover, by Lemma 0.2, we have paths

$$t_1(w) : (\mathsf{pr}_1 w, \mathsf{pr}_2 w) = w, \qquad t_2(w) : (\mathsf{pr}_1\mathsf{pr}_1 w, \mathsf{pr}_2\mathsf{pr}_1 w) = \mathsf{pr}_1 w,$$
$$s_1(v) : (\mathsf{pr}_1 v, \mathsf{pr}_2 v) = v, \qquad s_2(v) : (\mathsf{pr}_1\mathsf{pr}_2 v, \mathsf{pr}_2\mathsf{pr}_2 v) = \mathsf{pr}_2 v,$$

so that

$$\mathsf{ap}_\alpha(t_2(w)) : ((\mathsf{pr}_1\mathsf{pr}_1 w, \mathsf{pr}_2\mathsf{pr}_1 w), \mathsf{pr}_2 w) = (\mathsf{pr}_1 w, \mathsf{pr}_2 w),$$
$$\mathsf{ap}_\beta(s_2(v)) : (\mathsf{pr}_1 v, (\mathsf{pr}_1\mathsf{pr}_2 v, \mathsf{pr}_2\mathsf{pr}_2 v)) = (\mathsf{pr}_1 v, \mathsf{pr}_2 v),$$

where $\alpha :\equiv \lambda a.(a, \mathsf{pr}_2 w)$ and $\beta :\equiv \lambda b.(\mathsf{pr}_1 v, b)$. Therefore,

$$\mathsf{ap}_\alpha(t_2(w)) \centerdot t_1(w) : f(g(w)) = w,$$
$$\mathsf{ap}_\beta(s_2(v)) \centerdot s_1(v) : g(f(v)) = v.$$

So, we have constructed homotopies $h : f \circ g \sim \mathsf{id}_N$ and $k : g \circ f \sim \mathsf{id}_M$ given by

$$h(w) :\equiv \mathsf{ap}_\alpha(t_2(w)) \centerdot t_1(w), \qquad k(v) :\equiv \mathsf{ap}_\beta(s_2(v)) \centerdot s_1(v). \qquad \square$$

Lemma 2.4 ($\sum$-types are "commutative")**.** *For all $A : \mathcal{U}$, $B : A \to \mathcal{U}$, and $C : \left(\sum_{(a:A)} B(a)\right) \to \mathcal{U}$,*

$$\sum_{(a:A)} \sum_{(b:B)} C((a,b)) \simeq \sum_{(b:B)} \sum_{(a:A)} C((a,b)). \tag{2.4}$$

Proof. Let M and N denote the left and the right sides of (2.4), respectively. The idea of the proof is simply to define $f : M \to N$ and $g : N \to M$ so that

$$f((a,(b,c))) \equiv (b,(a,c)), \qquad g((b,(a,c))) \equiv (a,(b,c)),$$

and hence

$$f(g((b,(a,c))) \equiv (b,(a,c)), \qquad g(f((a,(b,c)))) \equiv (a,(b,c)),$$

for all $a : A$, $b : B(a)$, and $c : B((a,b))$. These conditions hold for

$$f :\equiv \lambda(v : M).(\mathsf{pr}_1\mathsf{pr}_2 v, (\mathsf{pr}_1 v, \mathsf{pr}_2\mathsf{pr}_2 v)),$$
$$g :\equiv \lambda(w : N).(\mathsf{pr}_1\mathsf{pr}_2 w, (\mathsf{pr}_1 w, \mathsf{pr}_2\mathsf{pr}_2 w)).$$

We leave to the reader the task of constructing homotopies $h : f\circ g \sim \mathsf{id}_N$ and $h : g\circ f \sim \mathsf{id}_M$ as we did in Lemma 2.3. $\square$

The next lemma is a version of (Univalent Foundations Program, 2013, Lemma 4.2.3) in a more direct language.

Lemma 2.5. *Given $(f,(g,(h,k))) : A \simeq B$, there is a $h' : f \circ g \sim \mathsf{id}_B$ such that, for any $x : A$, $\mathsf{ap}_f(k(x)) = h'(f(x))$.*

Proof. Define $h' :\equiv \lambda(y : B).h(f(g(y)))^{-1} \centerdot (\mathsf{ap}_f(k(g(y))) \centerdot h(y))$. We need to prove that

$$\mathsf{ap}_f(k(x)) = h(f(g(f(x))))^{-1} \centerdot (\mathsf{ap}_f(k(g(f(x)))) \centerdot h(f(x))), \tag{2.5}$$

for any $x : A$. Applying Corollary 1.13 for $g \circ f : A \to A$ and $k : g \circ f \sim \mathsf{id}_A$, we have a path $p : k(g(f(x))) = \mathsf{ap}_{g\circ f}(k(x))$. Setting $\mu :\equiv \lambda s.\, \mathsf{ap}_f(s) \centerdot h(f(x))$, it follows that

$$\mathsf{ap}_\mu(p) : \mathsf{ap}_f(k(g(f(x)))) \centerdot h(f(x)) = \mathsf{ap}_f(\mathsf{ap}_{g\circ f}(k(x))) \centerdot h(f(x)).$$

But, by (Lemma 1.4(c))$^{-1}$, there is a $q : \mathsf{ap}_f(\mathsf{ap}_{g\circ f}(k(x))) = \mathsf{ap}_{f\circ g\circ f}(k(x))$ so that

$$\mathsf{ap}_\nu(q) : \mathsf{ap}_f(\mathsf{ap}_{g\circ f}(k(x))) \centerdot h(f(x)) = \mathsf{ap}_{f\circ g\circ f}(k(x)) \centerdot h(f(x)),$$

for $\nu :\equiv \lambda s.\, s \centerdot h(f(x))$. Moreover, by replacing (simultaneously) H with h, x with $f(g(f(x)))$, g with f, p with $k(x)$, and f with $f \circ g \circ f$ in (1.1), we obtain a path

$$r : h(f(g(f(x)))) \centerdot \mathsf{ap}_f(k(x)) = \mathsf{ap}_{f\circ g\circ f}(k(x)) \centerdot h(f(x)).$$

So, we can see that $(\mathsf{ap}_\mu(p) \centerdot \mathsf{ap}_\nu(q)) \centerdot r^{-1}$ is an inhabitant of the type

$$\mathsf{ap}_f(k(g(f(x)))) \centerdot h(f(x)) = h(f(g(f(x)))) \centerdot \mathsf{ap}_f(k(x)).$$

Concatenating both sides of this equality with $h(f(g(f(x))))^{-1}$ to the left, it follows that

$$h(f(g(f(x))))^{-1} \centerdot (\mathsf{ap}_f(k(g(f(x)))) \centerdot h(f(x))) = \mathsf{ap}_f(k(x)),$$

by Lemma 1.2. Hence, (2.5) is true. □

2.2 Function extensionality and univalence

The next lemma tells that equality between non-dependent functions is a particular case of homotopy.

Lemma 2.6. *For all $A, B : \mathcal{U}$, $f, g : A \to B$, and $p : f = g$, there is a certain homotopy* $\mathsf{happly}(p)$ *from f to g. The function* happly *defined in this way satisfies the computation rule* $\mathsf{happly}(\mathsf{refl}_f) \equiv \lambda(x : A).\mathsf{refl}_{f(x)}$.

Proof. Define happly by

$$\mathsf{happly}(p) :\equiv \lambda(x : A).\mathsf{ap}_{e(x)}(p),$$

where $e(x) :\equiv \lambda(h : A \to B).h(x)$. (It is also easy to define such a homotopy by path induction.) □

As we commented at the beginning of Section 1.4, under the type-theoretic rules so far presented, we cannot deduce that any two homotopic functions are equal. But we can assert this as an axiom.

Axiom 2.1 (Function extensionality)**.** *The function* happly *defined in Lemma 2.6 has a quasi-inverse*

$$\mathsf{funext} : (f \sim g) \to (f = g).$$

The proof of the part (a) of the next lemma (which is used in the proofs of Lemmas 4.3, 5.2, and Theorem 4.8) uses the axiom of function extensionality.

Lemma 2.7. *Let* P *and* Q *be type families over* A*.*

(a) If $\prod_{(x:A)} P(x) \simeq Q(x)$*, then* $\left(\prod_{(x:A)} P(x)\right) \simeq \left(\prod_{(x:A)} Q(x)\right)$*.*

(b) If $\prod_{(x:A)} P(x) \simeq Q(x)$*, then* $\left(\sum_{(x:A)} P(x)\right) \simeq \left(\sum_{(x:A)} Q(x)\right)$*.*

Proof. Suppose that $\alpha : \prod_{(x:A)} P(x) \simeq Q(x)$.

(a) Let C and D denote $\prod_{(x:A)} P(x)$ and $\prod_{(x:A)} Q(x)$, respectively. We need to construct an element $(f, (g, (h, k)))$ of $C \simeq D$. In particular, we must give, for each $c : C$, a dependent function $f(c) : \prod_{(x:A)} Q(x)$. Note that, for any $x : A$, we can get an element of $Q(x)$ by applying $\mathsf{pr}_1(\alpha(x)) : P(x) \to Q(x)$ on $c(x) : P(x)$. So, we may define

$$f(c) :\equiv \lambda(x : A).\mathsf{pr}_1(\alpha(x), c(x)).$$

Similarly, one verifies that the definition

$$g(d) :\equiv \lambda(x : A).\mathsf{pr}_2(\alpha(x), d(x))$$

is suitable, for each $d : D$. So, we have constructed $f : C \to D$ and $g : D \to C$. Now, set $\mathsf{pr}_3 :\equiv \mathsf{pr}_1 \circ \mathsf{pr}_2 \circ \mathsf{pr}_2$, $\mathsf{pr}_4 :\equiv \mathsf{pr}_2 \circ \mathsf{pr}_2 \circ \mathsf{pr}_2$, and note that

$$\mathsf{pr}_3(\alpha(x), d(x)) : \mathsf{pr}_1(\alpha(x), \mathsf{pr}_2(\alpha(x), d(x))) = d(x),$$
$$\mathsf{pr}_4(\alpha(x), c(x)) : \mathsf{pr}_2(\alpha(x), \mathsf{pr}_1(\alpha(x), c(x))) = c(x),$$

for all $c : C$, $d : D$, and $x : A$. Since $f(g(d)) \equiv \lambda(x : A).\mathsf{pr}_1(\alpha(x), \mathsf{pr}_2(\alpha(x), d(x)))$ and $g(f(c)) \equiv \lambda(x : A).\mathsf{pr}_2(\alpha(x), \mathsf{pr}_1(\alpha(x), c(x)))$, it follows by $\lambda(x : A).c(x) \equiv c$ and $d \equiv \lambda(x : A).d(x)$ that

$$\mathsf{funext}(\lambda(x : A).\mathsf{pr}_3(\alpha(x), d(x))) : f(g(d)) = d,$$
$$\mathsf{funext}(\lambda(x : A).\mathsf{pr}_4(\alpha(x), c(x))) : g(f(c)) = c.$$

Thus, we may define $h : f \circ g \sim \mathsf{id}_D$ and $k : g \circ f \sim \mathsf{id}_C$ by

$$h(d) :\equiv \mathsf{funext}(\lambda(x : A).\mathsf{pr}_3(\alpha(x), d(x))),$$
$$k(c) :\equiv \mathsf{funext}(\lambda(x : A).\mathsf{pr}_4(\alpha(x), c(x))).$$

(b) In the same spirit of part (a), let V and W denote $\sum_{(x:A)} P(x)$ and $\sum_{(x:A)} Q(x)$, respectively. It is easy to see that $f : V \to W$ and $g : W \to V$ may be defined by

$$f(v) :\equiv (\mathsf{pr}_1(v), \mathsf{pr}_1(\alpha, \mathsf{pr}_2(v))),$$
$$g(w) :\equiv (\mathsf{pr}_1(w), \mathsf{pr}_2(\alpha, \mathsf{pr}_2(w))).$$

Moreover,

$$\begin{aligned} f(g(w)) &\equiv (\mathsf{pr}_1(g(w)), \mathsf{pr}_1(\alpha, \mathsf{pr}_2(g(w)))) \\ &\equiv (\mathsf{pr}_1(w), \mathsf{pr}_1(\alpha, \mathsf{pr}_2(\alpha, \mathsf{pr}_2(w)))) \\ &= (\mathsf{pr}_1(w), \mathsf{pr}_2(w)) && \text{by } p \\ &= w, && \text{by } q \end{aligned}$$

where $p :\equiv \mathsf{ap}_\mu(\mathsf{pr}_3(\alpha, \mathsf{pr}_2(w)))$, for $\mu :\equiv \lambda y.(\mathsf{pr}_1(w), y)$, and $q :\equiv \mathsf{uppt}(w)$. (Recall that $\mathsf{pr}_3 :\equiv \mathsf{pr}_1 \circ \mathsf{pr}_2 \circ \mathsf{pr}_2$.) Analogously, one proves $g(f(v)) = v$. □

The axiom of function extensionality is a consequence of another important axiom, called the *axiom of univalence*.

Lemma 2.8. *For all $A, B : \mathcal{U}$ and $p : A =_{\mathcal{U}} B$, there is a certain equivalence*

$$\mathsf{idtoeqv}(p) : A \simeq B.$$

The function idtoeqv *defined in this way satisfies the computation rule*

$$\mathsf{idtoeqv}(\mathsf{refl}_A) \equiv (\mathsf{id}_A, (\mathsf{id}_A, (\lambda a.\mathsf{refl}_a, \lambda a.\mathsf{refl}_a))).$$

Proof. Consider the function $\mathsf{transport}^{X \mapsto X}$, where $X \mapsto X$ denotes the type family that sends each type $X : \mathcal{U}$ to itself. It assigns to each $p : A = B$ a function $\mathsf{transport}^{X \mapsto X}(p)$ which has a quasi-inverse $\mathsf{transport}^{X \mapsto X}(p^{-1})$, by Example 2.3. These mutual quasi-inverses, together with the respective homotopies, give an equivalence from A to B. We define $\mathsf{idtoeqv}(p)$ as being this equivalence. The verification of the computation rule is immediate. Another easy way of proving this lemma is by applying path induction on p. □

Axiom 2.2 (Univalence)**.** *The function* idtoeqv *defined in Lemma 2.8 has a quasi-inverse (whose first coordinate is)*

$$\mathsf{ua} : (A \simeq B) \to (A =_{\mathcal{U}} B).$$

The axiom of univalence formalizes the common mathematical practice of identifying isomorphic structures.

By the homotopies involved in Axiom 2.2, we have

$$\mathsf{idtoeqv}(\mathsf{ua}(v)) = v, \tag{2.6}$$
$$p = \mathsf{ua}(\mathsf{idtoeqv}(p)), \tag{2.7}$$

for all $v : A \simeq B$ and $p : A = B$. Note that if q is an element of (2.6), then $\mathsf{happly}(\mathsf{ap}_{\mathsf{pr}_1}(q), a)$ is an inhabitant of

$$\mathsf{transport}^{X \mapsto X}(\mathsf{ua}(v), a) = \mathsf{pr}_1(v, a), \tag{2.8}$$

for all $a : A$, since $\mathsf{transport}^{X \mapsto X}(\mathsf{ua}(v)) \equiv \mathsf{pr}_1(\mathsf{idtoeqv}(\mathsf{ua}(v)))$.

3 Identity Types of Σ- and Π-types

In this chapter, we characterize some types $\mathsf{Id}_A(x, y)$, with A being of the form $\sum_{(z:B)} P(z)$ or the form $\prod_{(z:B)} P(z)$. Some lemmas about certain forms of transport are required for this purpose. We have put together these and other similar lemmas into Section 3.1. In Section 3.3, we interpret the first projection associated to any type family as a fibration.

3.1 Specific transport lemmas

In Section 1.3, we proved some lemmas about transport with respect to completely arbitrary type families. Now, we study transport with respect to more specific type families.

Lemma 3.1. *For any type B and every path $p : x =_A y$, there is a certain dependent function*

$$\mathsf{transpconst}_p^B : \prod_{b:B} \mathsf{transp}^{(\lambda(a:A).B)}(p, b) = b.$$

The map $p \mapsto \mathsf{transpconst}_p^B$ so defined satisfies $\mathsf{transpconst}^B_{\mathsf{refl}_x} \equiv \lambda(b : B).\mathsf{refl}_b$, for all $x : A$.

Proof. In the case when $x \equiv y$, we have $\mathsf{transp}^{(\lambda(x:A).B)}(\mathsf{refl}_x, b) \equiv \mathsf{id}_B(b) \equiv b$ for all $b : B$, so that we may define $\mathsf{transpconst}^B_{\mathsf{refl}_x} :\equiv \lambda(b : B).\mathsf{refl}_b$. The general assertion follows immediately by path induction. □

Lemma 3.2. *Let A be a type, $x_1, x_2, a : A$, and $p : x_1 = x_2$.*

(a) For all $q : a = x_1$, $\mathsf{transp}^{x\mapsto(a=x)}(p, q) = q \centerdot p$.

(b) For all $q : x_1 = a$, $\mathsf{transp}^{x\mapsto(x=a)}(p, q) = p^{-1} \centerdot q$.

(c) For all $q : x_1 = x_1$, $\mathsf{transp}^{x\mapsto(x=x)}(p, q) = p^{-1} \centerdot q \centerdot p$.

Proof. For the first two items, use path induction on p and based path induction on q. Recall that based path induction is very similar to ordinary path induction, it says that, fixed $a : A$, we prove a claim about every path $q : a =_A x$ simply by proving the trivial case when $x \equiv a$ and $q \equiv \mathsf{refl}_a$. Of course, given $a : A$, any claim about every path $q' : x =_A a$ can be converted into a logically equivalent claim about every path $q : a =_A x$, where $q \equiv p^{-1}$ (see Example 2.1). For item (c), it suffices to apply induction on p. □

Lemma 3.3. *Suppose given a type X, type families $A, B : X \to \mathcal{U}$, and a path $p : x_1 =_X x_2$. Let $A \to B$ denote the element of $X \to \mathcal{U}$ defined by $(A \to B)(x) :\equiv A(x) \to B(x)$, for*

all $x \in X$. For any $f : (A \to B)(x_1)$, we have an element $\mathsf{transpfun}_p^{A\to B}(f)$ of the type

$$\mathsf{transp}^{A\to B}(p, f) = \lambda(a : A(x_2)).\mathsf{transp}^B(p, f(\mathsf{transp}^A(p^{-1}, a))).$$

More concisely,

$$\mathsf{transpfun}_p^{A\to B}(f) : p_*(f) = \lambda a.p_*(f({p^{-1}}_*(a))).$$

In addition, the map $p \mapsto \lambda(f : (A \to B)(x_1)).\mathsf{transpfun}_p^{A\to B}(f)$ so defined is such that, for all $f : (A \to B)(x_1)$, $\mathsf{transpfun}_{\mathsf{refl}_x}^{A\to B}(f)$ is the constant path on $\mathsf{transp}^{A\to B}(\mathsf{refl}_x, f)$.

Proof. Suppose $x_2 \equiv x_1$. Setting $x :\equiv x_1 \equiv x_2$, we have, for each $f : (A \to B)(x)$,

$$\begin{aligned}
\mathsf{transp}^P(\mathsf{refl}_x, f) &\equiv \mathsf{id}_{A(x)\to B(x)}(f) \\
&\equiv f \\
&\equiv \lambda(a : A(x)).f(a) \\
&\equiv \lambda(a : A(x)).\mathsf{transp}^B(\mathsf{refl}_x, f(a)) \\
&\equiv \lambda(a : A(x)).\mathsf{transp}^B(\mathsf{refl}_x, f(\mathsf{transp}^A(\mathsf{refl}_x, a))).
\end{aligned}$$

Then, the constant path on $\mathsf{transp}^{A\to B}(\mathsf{refl}_x, f)$ is an inhabitant of

$$\mathsf{transp}^{A\to B}(\mathsf{refl}_x, f) = \lambda(a : A(x)).\mathsf{transp}^B(\mathsf{refl}_x, f(\mathsf{transp}^A(\mathsf{refl}_x^{-1}, a))).$$

The general statement follows immediately by path induction on p. □

See Lemma 3.10 for one more result about transport, which would be placed here if it did not use the function $\mathsf{dpair}^{=}$, defined in the next section.

3.2 Characterizations of identity types

3.2.1 Dependent pair types and non-dependent function types

Theorem 3.4. *For all $w, w' : A \times B$,*

$$(w =_{A\times B} w') \simeq (\mathsf{pr}_1(w) =_A \mathsf{pr}_1(w')) \times (\mathsf{pr}_2(w) =_B \mathsf{pr}_2(w')). \tag{3.1}$$

Proof. Let C and D denote the left and right sides of (3.1), respectively. A function $f : C \to D$ is given immediately by path induction with the computation rule $f(\mathsf{refl}_w) = (\mathsf{refl}_{\mathsf{pr}_1(w)}, \mathsf{refl}_{\mathsf{pr}_2(w)})$. Now, we have to construct a $g : D \to C$. By path induction (on p and q), it is easy to verify that there is a certain function

$$g_1 : \prod_{(x,x':A)} \prod_{(p:x=x')} \prod_{(y,y':B)} \prod_{(q:y=y')} (x, y) = (x', y'),$$

which corresponds to a

$$g_2 : \prod_{(x:A)} \prod_{(y,:B)} \prod_{(x':A)} \prod_{(y':B)} \prod_{(p:x=x')} \prod_{(q:y=y')} (x,y) = (x',y')$$
$$\equiv \prod_{(x:A)} \prod_{(y,:B)} \prod_{(x':A)} \prod_{(y':B)} (x = x') \to (y = y') \to ((x,y) = (x',y'))$$

under the evident logical equivalence between the types of these two functions. By applying the induction principle for product types twice, g_2 becomes

$$g_3 : \prod_{w,w':A\times B} (\mathsf{pr}_1(w) = \mathsf{pr}_1(w')) \to (\mathsf{pr}_2(w) = \mathsf{pr}_2(w')) \to (w = w').$$

So, for any $w, w' : A \times B$, $g_3(w, w')$ produces

$$g : (\mathsf{pr}_1(w) = \mathsf{pr}_1(w')) \times (\mathsf{pr}_2(w) = \mathsf{pr}_2(w')) \to (w = w'),$$

by the recursion principle for product types. Moreover, g_1, g_2, g_3, and g satisfy the computation rules

$$g((p,q)) \equiv g_3(w,w',p,q),$$
$$g_3((x,y),(x',y'),p,q) \equiv g_2(x,y,x',y',p,q),$$
$$g_2(x,y,x,y,\mathsf{refl}_x,\mathsf{refl}_y) \equiv g_1(x,x,\mathsf{refl}_x,y,y,\mathsf{refl}_y) \equiv \mathsf{refl}_{(x,y)}.$$

To prove that $f \circ g \sim \mathsf{id}_D$ and $g \circ f \sim \mathsf{id}_C$, consider $x :\equiv \mathsf{pr}_1(w)$, $y :\equiv \mathsf{pr}_2(w)$, $x' :\equiv \mathsf{pr}_1(w')$, and $y' :\equiv \mathsf{pr}_2(w')$. Given $d : D$, let d_1 and d_2 denote $\mathsf{pr}_1(d)$ and $\mathsf{pr}_2(d)$, respectively. On one hand, we have, by Lemma 0.2:

$$f(g(d)) = f(g((d_1,d_2))) \equiv f(g_3(w,w',d_1,d_2))$$
$$= f(g_3((x,y),(x',y'),d_1,d_2)) \equiv f(g_2(x,y,x',y',d_1,d_2)).$$

On the other hand, by the principle of path induction, we may assume that $x \equiv x'$, $y \equiv y'$, and $d \equiv \mathsf{refl}_{(x,y)}$, so that

$$f(g_2(x,y,x',y',d_1,d_2)) \equiv f(g_2(x,y,x,y,\mathsf{refl}_x,\mathsf{refl}_y)) \equiv f(g_1(x,x,\mathsf{refl}_x,y,y,\mathsf{refl}_y))$$
$$\equiv f(\mathsf{refl}_{(x,y)}) \equiv (\mathsf{refl}_x,\mathsf{refl}_y) \equiv (\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{refl}_{(x,y)}),\mathsf{ap}_{\mathsf{pr}_2}(\mathsf{refl}_{(x,y)}))$$
$$= \mathsf{refl}_{(x,y)} \equiv d,$$

by $\mathsf{uppt}(\mathsf{refl}_{(x,y)})$ (Lemma 0.2). Thus, $f \circ g \sim \mathsf{id}_D$. Finally, to see that $g \circ f \sim \mathsf{id}_C$, it suffices to note that if $x \equiv x'$ and $y \equiv y'$, then $g(f(\mathsf{refl}_{(x,y)})) \equiv g((\mathsf{refl}_x,\mathsf{refl}_y)) \equiv \mathsf{refl}_{(x,y)}$. □

Definition 3.1. Given $w, w' : A \times B$, we denote by $\mathsf{pair}^=$ the element of

$$\prod_{(p:\mathsf{pr}_1(w)=\mathsf{pr}_1(w'))} \prod_{(q:\mathsf{pr}_2(w)=\mathsf{pr}_2(w'))} w = w'$$

defined so that $\mathsf{pair}^=(p,q)$ is the path from w to w' that corresponds to (p,q) under the equivalence from Theorem 3.4. Thus, $\mathsf{pair}^=(\mathsf{refl}_a,\mathsf{refl}_b) = \mathsf{refl}_{(a,b)}$, for any $a : A$ and $b : B$.

Now we generalize Theorem 3.4 for dependent pair types.

Theorem 3.5. *For every type family $P : A \to \mathcal{U}$ and any $w, w' : \sum_{(x:A)} P(x)$, we have*

$$(w = w') \simeq \sum_{p:\mathsf{pr}_1(w)=\mathsf{pr}_1(w')} \mathsf{transp}^P(p, \mathsf{pr}_2(w)) = \mathsf{pr}_2(w'). \tag{3.2}$$

Proof. Let $S(w, w')$ denote the right side of (3.2), for each w and w' having the type $T :\equiv \sum_{(x:A)} P(x)$. Fix $v : T$. We have to construct, for each $w : T$, two mutual quasi-inverses $f_w : (v = w) \to S(v, w)$ and $g_w : S(v, w) \to (v = w)$.

Consider the map $c :\equiv \lambda(w : T).(\mathsf{refl}_{\mathsf{pr}_1(w)}, \mathsf{refl}_{\mathsf{pr}_2(w)}) : \prod_{(w:T)} C(w, w, \mathsf{refl}_w)$, where $C : \prod_{(w,w':T)} \prod_{(r:w=w')} \mathcal{U}$ is defined by $C(w, w', r) :\equiv S(w, w')$. We have,

$$\mathsf{ind}_{=_T}(C, c) : \prod_{(w,w':T)} \prod_{(r:w=w')} C(w, w', r)$$

so that we may define

$$f_w :\equiv \mathsf{ind}_{=_T}(C, c, v, w) : \prod_{r:v=w} C(v, w, r) \equiv ((v = w) \to S(v, w)).$$

Now, observe that the type

$$\prod_{(x,x':A)} \prod_{(p:x=x')} \prod_{(y:P(x))} \prod_{(y':P(x'))} \prod_{(q:p_*(y)=y')} (x, y) = (x', y') \tag{3.3}$$

is inhabited. The verification is straightforward by path induction on p and based path induction on q. Furthermore, (3.3) is logically equivalent to

$$\prod_{(x,x':A)} \prod_{(y:P(x))} \prod_{(y':P(x'))} \prod_{(p:x=x')} \prod_{(q:p_*(y)=y')} (x, y) = (x', y').$$

So, for all $x, x' : A$, $y : P(x)$, and $y' : P(x')$, the type $\prod_{(p:x=x')} \prod_{(q:p_*(y)=y')} (x, y) = (x', y')$ is inhabited. In particular, for each $w : T$, we have an element of the type

$$\prod_{(p:\mathsf{pr}_1(v)=\mathsf{pr}_1(w))} \prod_{(q:p_*(\mathsf{pr}_2(v))=\mathsf{pr}_2(w))} (\mathsf{pr}_1(v), \mathsf{pr}_2(v)) = (\mathsf{pr}_1(w), \mathsf{pr}_2(w))$$

which is transformed into an inhabitant of

$$S(v, w) \to ((\mathsf{pr}_1(v), \mathsf{pr}_2(v)) = (\mathsf{pr}_1(w), \mathsf{pr}_2(w))),$$

by the recursion principle for dependent pair types. But, by Lemmas 0.2 and 2.2,

$$((\mathsf{pr}_1(v), \mathsf{pr}_2(v)) = (\mathsf{pr}_1(w), \mathsf{pr}_2(w))) \simeq (v = w).$$

Therefore, we have a

$$g_w : S(v, w) \to (v = w).$$

It remains to show that $g_w : \mathsf{qinv}(f_w)$, for any $w : T$. For this purpose, define $v_1 :\equiv \mathsf{pr}_1(v)$, $v_2 :\equiv \mathsf{pr}_2(v)$, and verify that

$$f_v(\mathsf{refl}_v) = f_{(v_1,v_2)}(\mathsf{refl}_{(v_1,v_2)}) \equiv (\mathsf{refl}_{v_1}, \mathsf{refl}_{v_2}),$$
$$g_v((\mathsf{refl}_{v_1}, \mathsf{refl}_{v_2})) = g_{(v_1,v_2)}((\mathsf{refl}_{v_1}, \mathsf{refl}_{v_2})) \equiv \mathsf{refl}_{(v_2,v_2)} = \mathsf{refl}_v,$$

so that

$$g_v(f_v(\mathsf{refl}_v)) = g_v((\mathsf{refl}_{v_1}, \mathsf{refl}_{v_2})) = \mathsf{refl}_v,$$
$$f_v(g_v((\mathsf{refl}_{v_1}, \mathsf{refl}_{v_2}))) = f_v(\mathsf{refl}_v) = (\mathsf{refl}_{v_1}, \mathsf{refl}_{v_2}).$$

By based path induction, it follows that

$$g_w(f_w(r)) = r, \tag{3.4}$$
$$f_w(g_w((p,q))) = (p,q), \tag{3.5}$$

for all $w : T$, $r : v = w$, $x : A$, $p : v_1 = x$, $y : P(x)$, and $q : p_*(x) = y$. From (3.4), we have that $g_w \circ f_w \sim \mathsf{id}_{(v=w)}$, for any $w : T$. And from (3.5), by induction on $S(v,w)$, we see that $f_w \circ g_w \sim \mathsf{id}_{S(v,w)}$, for all $w : T$. □

Definition 3.2. Given $w, w' : \sum_{(x:A)} P(x)$, we denote by $\mathsf{dpair}^{=}$ the element of

$$\prod_{(p:\mathsf{pr}_1(w)=\mathsf{pr}_1(w'))} \prod_{(q:p_*(\mathsf{pr}_2(w))=\mathsf{pr}_2(w'))} w = w'$$

defined so that $\mathsf{dpair}^{=}(p,q)$ is the path from w to w' that corresponds to (p,q) under the equivalence from Theorem 3.5. Thus, $\mathsf{dpair}^{=}(\mathsf{refl}_a, \mathsf{refl}_b) = \mathsf{refl}_{(a,b)}$, for any $a : A$ and $b : P(a)$.

Recall that in Lemma 1.5 we considered a path between points $u : B(x)$ and $v : B(y)$ of different types as being a path from (x,u) to (y,v) in $\sum_{(a:A)} B(a)$. Now, by Theorem 3.5, we see that such a path from (x,u) to (y,v) corresponds to a pair (p,q) with $p : x = y$ and $q : p_*(u) = v$. This characterization of paths between points of different types is more descriptive, so it motivates another version of Lemma 1.5.

Definition 3.3. Given a type family $B : A \to \mathcal{U}$ and a path $p : x =_A y$, a *dependent path* from $u : B(x)$ to $v : B(y)$ lying over p is an element of the type

$$(u =^B_p v) :\equiv (\mathsf{transp}^B(p,u) =_{B(y)} v).$$

Lemma 3.6. *For all $f : \prod_{(a:A)} B(a)$, there is a function apd_f that assigns, to each $p : x =_A y$, a path $\mathsf{apd}_f(p) : f(x) =^B_p f(y)$ in such a way that $\mathsf{apd}_f(\mathsf{refl}_a) \equiv \mathsf{refl}_{f(a)}$ for all $a : A$.*

Proof. Immediate by path induction on p. □

Compare Lemmas 3.6 and 1.5.

The following two lemmas are simple path inductions involving $\mathsf{dpair}^{=}$ which are used in the proof of Lemma 5.2.

Lemma 3.7. *Let $P : A \to \mathcal{U}$. For all $p_1 : x =_A y$, $q_1 : y =_A z$, $p_2 : \mathsf{transp}^P(p_1, u) =_{P(y)} v$, and $q_2 : \mathsf{transp}^P(q_1, v) =_{P(z)} w$,*

$$\mathsf{dpair}^{=}(p_1, p_2) \centerdot \mathsf{dpair}^{=}(q_1, q_2) = \mathsf{dpair}^{=}(p_1 \centerdot q_1,\ p_2 \centerdot q_2).$$

Proof. Immediate by path induction on p_1, p_2, q_1, and q_2. □

Lemma 3.8. *Let P be a type family over A. Given $a : A$, if $f : P(a) \to \sum_{(x:A)} P(x)$ is defined by $f(u) :\equiv (a, u)$ and p is any path in A from a to itself, then, for all $q : \mathsf{transp}^P(p, u) =_{P(a)} v$,*

$$\mathsf{ap}_f(q) = \mathsf{dpair}^{=}(\mathsf{refl}_a, q).$$

Proof. Fix $u : P(a)$ and apply based path induction on q. □

We end this section with a characterization of types of dependent paths between non-dependent functions. In Lemma 3.11, we generalize it for dependent functions.

Lemma 3.9. *Given $A, B : X \to \mathcal{U}$, let $A \to B$ denote the element of $X \to \mathcal{U}$ defined by $(A \to B)(x) :\equiv A(x) \to B(x)$, for all $x \in X$. For any $p : x_1 =_X x_2$, $f : (A \to B)(x_1)$, and $g : (A \to B)(x_2)$, we have:*

$$\left(\mathsf{transp}^{A\to B}(p, f) = g\right) \simeq \prod_{a:A(x_1)} \left(\mathsf{transp}^{B}(p, f(a)) = g(\mathsf{transp}^{A}(p, a))\right).$$

More concisely,

$$(p_*(f) = g) \simeq \prod_{a:A(x_1)} p_*(f(a)) = g(p_*(a)). \tag{3.6}$$

In addition, if $q : p_*(f) = g$ corresponds to q' under this equivalence, then, for all $a : A(x_1)$, the path

$$\mathsf{happly}(q, p_*(a)) : (p_*(f))(p_*(a)) = g(p_*(a))$$

is equal to the composite

$$\begin{aligned}
(p_*(f))(p_*(a)) &= p_*\left(f({p^{-1}}_*(p_*(a)))\right) && \text{by } \mathsf{ap}_e(\mathsf{transpfun}_p^{A\to B}(f)) \\
&= p_*(f(a)) && \text{by } \mathsf{ap}_{p_*\circ f}(\mathsf{ittranspinv}_1(p, a)) \\
&= g(p_*(a)), && \text{by } q'(a)
\end{aligned}$$

where $e \equiv \lambda(f : P(x_2)).f(p_*(a))$.

Proof. In the trivial case (when $x_1 \equiv x_2$ and $p \equiv \mathsf{refl}_{x_1}$), (3.6) reduces to the extensionality axiom. So, by path induction, (3.6) is true in generality. The second part of the lemma is also proved by path induction, using the fact that, in the trivial case, q' is judgementally equal to $\mathsf{happly}(q)$, whereas $\mathsf{ap}_e(\mathsf{transpfun}_p^{A\to B}(f))$ and $\mathsf{ap}_{p_*\circ f}(\mathsf{ittranspinv}_1(p,a))$ reduce to reflexivities. □

3.2.2 Dependent function types

Lemma 3.10. *Suppose given a type X, a path $p : x_1 =_X x_2$, and type families $A : X \to \mathcal{U}$ and $B : \prod_{(x:X)} A(x) \to \mathcal{U}$. Consider $\Pi_A(B) : X \to \mathcal{U}$ and $\widehat{B} : \left(\sum_{(x:X)} A(x)\right) \to \mathcal{U}$ defined by*

$$\Pi_A(B)(x) :\equiv \prod_{a:A(x)} B(x,a), \tag{3.7}$$

$$\widehat{B}(w) :\equiv B(\mathsf{pr}_1(w), \mathsf{pr}_2(w)). \tag{3.8}$$

For all $f : \Pi_A(B)(x_1)$, the type

$$\mathsf{transp}^{\Pi_A(B)}(p,f) = \lambda\,(a : A(x_2))\,.\mathsf{transp}^{\widehat{B}}\left(\left(\mathsf{dpair}^{=}\left(p^{-1},\, \mathsf{refl}_{{p^{-1}}_*(a)}\right)\right)^{-1},\ f\left({p^{-1}}_*(a)\right)\right)$$

is inhabited by certain path, which we denote by $\mathsf{transpdfun}_p^{\Pi_A(B)}(f)$. So, for any $a : A(x_2)$, we have an element $\mathsf{happly}(\mathsf{transpdfun}_p^{\Pi_A(B)}(f), a)$ of the type

$$p_*(f)(a) = \mathsf{dpair}^{=}\left(p^{-1},\, \mathsf{refl}_{{p^{-1}}_*(a)}\right)^{-1}_{*}\left(f({p^{-1}}_*(a))\right).$$

Proof. Immediate by path induction on p. □

Lemma 3.11. *Given type families $A, B : X \to \mathcal{U}$, let $\Pi_A(B) : \prod_{(x:X)} \to \mathcal{U}$ and $\widehat{B}$ be defined as in (3.7) and (3.8). For any $p : x_1 =_X x_2$, $f : \Pi_A(B)(x_1)$, and $g : \Pi_A(B)(x_2)$, we have:*

$$\left(\mathsf{transp}^{\Pi_A(B)}(p,f) = g\right) \simeq \prod_{a:A(x_1)} \left(\mathsf{transp}^{\widehat{B}}\left(\mathsf{dpair}^{=}\left(p,\, \mathsf{refl}_{p_*(a)}\right),\ f(a)\right) = g(p_*(a))\right).$$

More concisely,

$$(p_*(f) = g) \simeq \prod_{a:A(x_1)} \mathsf{dpair}^{=}\left(p,\, \mathsf{refl}_{p_*(a)}\right)_*(f(a)) = g(p_*(a)).$$

In addition, if $q : p_*(f) = g$ corresponds to q' under this equivalence, then, for all $a : A(x_1)$, the path

$$\mathsf{happly}(q, p_*(a)) : (p_*(f))(p_*(a)) = g(p_*(a))$$

is equal to the composite

$$\begin{aligned}
(p_*(f))(p_*(a)) &= \mathsf{dpair}^{=}\left(p^{-1}, \mathsf{refl}_{{p^{-1}}_*(p_*(a))}\right)^{-1}_{*}\left(f\left({p^{-1}}_*(p_*(a))\right)\right) && \text{by } p_1(a)\\
&= \mathsf{dpair}^{=}\left(p^{-1}, \mathsf{refl}_a\right)^{-1}_{*}(f(a)) && \text{by } p_2(a)\\
&= \mathsf{dpair}^{=}\left(p,\ \mathsf{refl}_{p_*(a)}\right)_{*}(f(a)) && \text{by } p_3(a)\\
&= g(p_*(a)), && \text{by } q'(a)
\end{aligned}$$

where

$$\begin{aligned}
p_1(a) &:\equiv \mathsf{ap}_{e_1}(\mathsf{transpfun}_p^{A\to B}(f))\\
&\qquad \textit{for } e_1 :\equiv \lambda(h : (A \to B)(x_2)).h(p_*(a)),\\
p_2(a) &:\equiv \mathsf{ap}_{e_2}(\mathsf{ittranspinv}_1(p, a))\\
&\qquad \textit{for } e_2 :\equiv \lambda(u : A(x_1)).\mathsf{dpair}^{=}\left(p^{-1}, \mathsf{refl}_u\right)^{-1}_{*}(f(u)),\\
p_3(a) &:\equiv \mathsf{ap}_{e_3}(r)\\
&\qquad \textit{for } e_3 :\equiv \lambda s.s_*(f(a))\\
&\qquad \textit{and } r : \mathsf{dpair}^{=}\left(p^{-1}, \mathsf{refl}_a\right)^{-1} = \mathsf{dpair}^{=}\left(p,\ \mathsf{refl}_{p_*(a)}\right) \textit{ given by induction on } p.
\end{aligned}$$

Proof. Similar to the proof of Lemma 3.9. □

3.3 Type families as fibrations

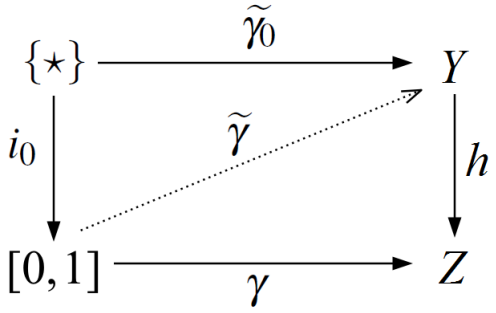

Figure 1 – Commutative diagram for the path lifting property. The symbol i_0 denotes the natural inclusion $i_0(\star) = 0$.

In homotopy theory, a map $h : Y \to Z$ between topological spaces is said to have the *path lifting property* if, for any path $\gamma : [0, 1] \to Y$ and any point $y_0 \in Y$ being a lift of $\gamma(0)$ (i.e., such that $f(y_0) = \gamma(0)$), there is a unique path $\widetilde{\gamma} : [0, 1] \to Y$ that starts in y_0 (i.e., $\widetilde{\gamma}(0) = y_0$) and is a lifting of γ (i.e., $h \circ \widetilde{\gamma} = \gamma$). (See Figure 1.) The points $\gamma(0) \in Z$ and $y_0 \in Y$ may be thought as constant maps $\gamma_0 : t \mapsto \gamma(0)$ and $\widetilde{\gamma_0} : t \mapsto y_0$ on a certain

unit set $\{\star\}$ into their respective spaces (Z and Y), so that the condition $h(y_0) = \gamma(0)$ may be interpreted as $h \circ \widetilde{\gamma_0} = \gamma_0$.

The next lemma tells that the first projection $\mathsf{pr}_1 : \left(\sum_{(x:A)} P(x)\right) \to A$ associated to any type family $P : A \to \mathcal{U}$ satisfies the path lifting property in the homotopy-theoretic interpretation of type theory. As in Lemma 3.6, a path between points $u : B(x)$ and $v : B(y)$ of different types is viewed as a pair (p, q), with $p : x = y$ and $q : u =^B_p v$.

Lemma 3.12 (Path lifting property for type families)**.** *Given a type family P over A, consider the first projection $\mathsf{pr}_1 : T \to A$, where $T :\equiv \sum_{(x:A)} P(x)$. For any $p : x =_A y$ and $u : P(x)$, we have a path*

$$\mathsf{lift}^P(p, u) : (x, u) =_T (y, p_*(u))$$

such that $\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{lift}^P(p, u)) = p$. The function lift^P so defined, sometimes abbreviated as lift, satisfies the computation rule

$$\mathsf{lift}^P(\mathsf{refl}_x, u) \equiv \mathsf{refl}_{(x,u)}$$

for all $x : A$ and $u : P(x)$. Moreover, fixed $p : x =_A y$ and $u : P(x)$, any path $\widetilde{p} : (x, u) =_T w$ such that $\mathsf{ap}_{\mathsf{pr}_1}(\widetilde{p}) = p$ is propositionally equal to $\mathsf{lift}^P(p, u)$.

Proof. Since $(x, u) \equiv (x, (\mathsf{refl}_x)_*(u))$, we have $\mathsf{refl}_{(x,u)} : (x, u) = (x, \mathsf{refl}_{x*}(u))$. From this, an easy path induction gives the required function lift with the required computation rule. Now,

$$\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{lift}(\mathsf{refl}_x, u)) \equiv \mathsf{ap}_{\mathsf{pr}_1}(\mathsf{refl}_{(x,u)}) \equiv \mathsf{refl}_{\mathsf{pr}_1((x,u))} \equiv \mathsf{refl}_x,$$

so that, by path induction, $\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{lift}(p, u)) = p$, for all $p : x =_A y$ and $u : P(x)$. It remains to prove the "uniqueness" of $\mathsf{lift}(p, u)$, which means to construct an element of the type

$$\prod_{(x,y:A)} \prod_{(p:x=y)} \prod_{(u:P(x))} \prod_{(w:T)} \prod_{(\widetilde{p}:(x,u)=w)} (\mathsf{ap}_{\mathsf{pr}_1}(\widetilde{p}) = p) \to (\widetilde{p} = \mathsf{lift}(p, u)). \tag{3.9}$$

This can be done easily by induction on p and based induction on $\widetilde{p}$, since the computation rules of $\mathsf{ap}_{\mathsf{pr}_1}$ and lift imply that $(\mathsf{refl}_x = \mathsf{refl}_x) \to (\mathsf{refl}_{(x,u)} = \mathsf{refl}_{(x,u)})$ is definitionally equal to $(\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{refl}_{(x,u)}) = \mathsf{refl}_x) \to (\mathsf{refl}_{(x,u)} = \mathsf{lift}(\mathsf{refl}_x, u))$ and hence

$$\lambda(r : \mathsf{refl}_x = \mathsf{refl}_x).\mathsf{id}_{(\mathsf{refl}_{(x,u)} = \mathsf{refl}_{(x,u)})} : \left((\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{refl}_{(x,u)}) = \mathsf{refl}_x) \to (\mathsf{refl}_{(x,u)} = \mathsf{lift}(\mathsf{refl}_x, u))\right).$$

More formally, let $C : \prod_{(x,y:A)} \prod_{(p:x=y)} \prod_{(u:P(x))} \prod_{(w:T)} \prod_{(\widetilde{p}:(x,u)=w)} \mathcal{U}$ be defined by

$$C(x, y, p, u, w, \widetilde{p}) :\equiv (\mathsf{ap}_{\mathsf{pr}_1}(\widetilde{p}) = p) \to (\widetilde{p} = \mathsf{lift}(p, u)).$$

As we saw above, for all $x : A$ and $u : P(x)$, the function

$$g :\equiv \lambda(r : \mathsf{refl}_x = \mathsf{refl}_x).\mathsf{id}_{(\mathsf{refl}_{(x,u)} = \mathsf{refl}_{(x,u)})}$$

has type $C_1((x,u),\mathsf{refl}_{(x,u)})$, where $C_1 :\equiv C(x,x,\mathsf{refl}_x,u)$. Therefore, by based path induction on $\widetilde{p}$ and lambda abstraction on u and x, the type

$$\prod_{(x:A)}\prod_{(u:P(x))}\prod_{(w:T)}\prod_{(\widetilde{p}:(x,u)=w)} C_1(w,\widetilde{p}) \tag{3.10}$$

is inhabited by

$$g_1 :\equiv \lambda(x:A).\lambda(u:P(x)).\mathsf{ind}'_{=_T}(C_1,g).$$

But (3.10) is definitionally equal to

$$\prod_{x:A} C_2(x,x,\mathsf{refl}_x), \tag{3.11}$$

where $C_2 : \prod_{(x,y:A)}\prod_{(p:x=y)}\prod_{(u:P(x))}\mathcal{U}$ is defined by

$$C_2(x,y,p) :\equiv \prod_{(u:P(x))}\prod_{(w:T)}\prod_{(\widetilde{p}:(x,u)=w)} C(x,y,p,u,w,\widetilde{p}).$$

Finally, by ordinary path induction, we obtain an element $\mathsf{ind}_{=_A}(C_2,g_1)$ of the type

$$\prod_{(x,y:A)}\prod_{(p:x=y)} C_2(x,y,p),$$

which is judgementally equal to (3.9). □

Given set-theoretic functions $h : A \to B$ and $g : B \to C$, a lift (or lifting) of h (across g) is a function $f : A \to C$ such that $g \circ f = h$. We say in this situation that f lifts (or is lifting[1]) h. The same nomenclature applies for morphisms in any category, which may be continuous maps, type-theoretic functions, etc.

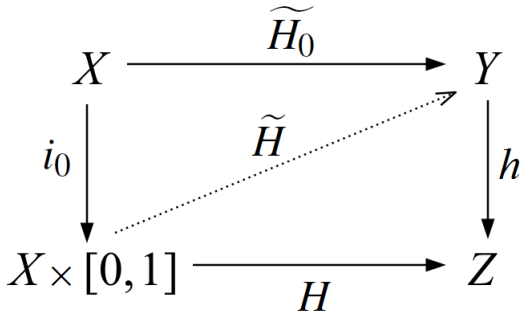

Figure 2 – Commutative diagram for the homotopy lifting property. The symbol i_0 denotes the natural inclusion $i_0(x) = (x,0)$.

Given a topological space X and a continuous map $h : Y \to Z$, we say that h satisfies the *homotopy lifting property* (with respect to X) if for any homotopy $H : X \times [0,1] \to Z$

[1] Other authors say that h lifts to f or h lifts across g. Our nomenclature is according to (HATCHER, 2002, p. 60).

and any continuous $\widetilde{H_0} : X \to Y$ lifting $H|_{X\times\{0\}}$ across h, there is a unique homotopy $\widetilde{H} : X \times [0,1] \to Z$ lifting H such that $\widetilde{H}|_{X\times\{0\}} = \widetilde{H_0}$ (see Figure 2). A (Hurewicz) *fibration* is a continuous map that satisfies the homotopy lifting property with respect to any space.

Notice that if we take X being any one point space space $\{\star\}$, the homotopy lifting property reduces to the path lifting property (via the identification $\{\star\} \times [0,1] \simeq [0,1]$ and other identifications mentioned at p. 3.3). Conversely, if a continuous map $h : Y \to Z$ satisfies the path lifting property, then h satisfies the homotopy lifting property with respect to any space X. Since we have proved that type families satisfy the path lifting property, *type families correspond to fibrations* in the homotopy interpretation of type theory.

4 Contractibility and Fiberwise Equivalences

4.1 Contractibility

Definition 4.1. A type A is said to be

(a) a *set* if it does not have nontrivial paths, i.e., if the type

$$\mathsf{isset}(A) :\equiv \prod_{(x,y:A)} \prod_{(p,q:x=y)} p = q$$

is inhabited.

(b) *contractible* if there is an element $a : A$, called the *center of contraction* of A, such that $a = x$ for all $x : A$. To put it another way, A is contractible if the type

$$\mathsf{iscontr}(A) :\equiv \sum_{(a:A)} \prod_{(x:A)} (a = x)$$

is inhabited.

Lemma 4.1 (Contractibility is a homotopy invariant)**.** *If $(f, (g, (h, k))) : A \simeq B$ and A is contractible with center a, then B is contractible with center $f(a)$.*

Proof. Since A is contractible with center a, there is a dependent function $c : \prod_{(x:A)}(a = x)$. For any $y : B$, since $c(g(y)) : a = g(y)$, we have $\mathsf{ap}_f(c(g(y))) : f(a) = f(g(y))$, so that

$$\mathsf{ap}_f(c(g(y))) \centerdot h(y) : f(a) = y. \qquad \square$$

Lemma 4.2. *Every contractible type is a set.*

Proof. Suppose that A is contractible. So, there is some $f : \prod_{(x:A)}(a = x)$, with $a : A$. Given $x_1, x_2 : X$, for all $p : x_1 = x_2$, we have:

$$\begin{aligned} f(x_1) \centerdot p &= \mathsf{transp}^{x \mapsto (a=x)}(p, f(x_1)) && \text{by (Lemma 3.2(a))}^{-1} \\ &= f(x_2). && \text{by } \mathsf{apd}_f(p) \end{aligned}$$

Denoting by q the above constructed element of $f(x_1) \centerdot p = f(x_2)$, it follows that

$$\begin{aligned} p &= \mathsf{refl}_{x_1} \centerdot p && \text{by Lemma 1.2(b)} \\ &= (f(x_1)^{-1} \centerdot f(x_1)) \centerdot p && \text{by } \mathsf{ap}_h((\text{Lemma 1.2(c)})^{-1}) \\ &= f(x_1)^{-1} \centerdot (f(x_1) \centerdot p) && \text{by (Lemma 1.2(f))}^{-1} \\ &= f(x_1)^{-1} \centerdot f(x_2), && \text{by } \mathsf{ap}_g(q) \end{aligned}$$

where $g :\equiv \lambda(r : a = x_2).f(x_1)^{-1} \centerdot r$ and $h :\equiv \lambda(r : x_1 = x_1).r \centerdot p$. Thus, all the elements of $x_1 = x_2$ are equal to $f(x_1)^{-1} \centerdot f(x_2)$. Hence, the elements of $x_1 = x_2$ are pairwise equal. $\square$

Lemma 4.3. *Let A be a type. For all $a : A$,*

(a) $\sum_{(x:A)}(a = x)$ *is contractible;*

(b) $\sum_{(x:A)}(x = a)$ *is contractible.*

Proof.

(a) We show that (a, refl_a) is the center of contraction of $\sum_{(x:A)}(a = x)$. Given $x : A$ and $p : a = x$,

$$\begin{aligned}
&((a, \mathsf{refl}_a) = (x, p)) \\
&\quad \simeq \sum_{q:a=x} \mathsf{transp}^{x \mapsto (a=x)}(p, \mathsf{refl}_a) = p && \text{by Theorem 3.5} \\
&\quad \simeq \sum_{q:a=x} \mathsf{refl}_a \centerdot p = p && \text{by Lemmas 3.2(a), 2.2, and 2.7(b)} \\
&\quad \simeq \sum_{q:a=x} p = p. && \text{by Lemmas } (1.2(\text{b}))^{-1}\text{, 2.2, and 2.7(b)}
\end{aligned}$$

The pair $(p, \mathsf{refl}_p) : \sum_{(q:a=x)}(p = p)$ corresponds under this equivalence to a path $g(x, p)$ between (a, refl_a) and (x, p). Thus, we have a function

$$g : \prod_{(x:A)} \prod_{(p:a=x)} (a, \mathsf{refl}_a) = (x, p),$$

which produces

$$f : \prod_{v:\sum_{(x:A)}(a=x)} (a, \mathsf{refl}_a) = v,$$

by the induction principle for dependent pair types.

(b) This is a simple consequence of part (a) and Lemmas 2.2, 2.7(b) and 4.1. Alternatively, the proof of part (a) can be easily adapted for this one, with the same center of contraction. □

Lemma 4.4. *Every map between contractible types is an equivalence.*

Proof. Let A and B be contractible types with centers of contraction $a : A$ and $b : B$. Given $f : A \to B$, define $g : B \to A$ by $g \equiv \lambda(x : B).a$. So, $g \circ f, \mathsf{id}_A : A \to A$ and $f \circ g, \mathsf{id}_B : B \to B$. Since A is contractible, we have: $\mathsf{id}_A \sim \lambda(x : A).a$. From this, by Lemma 1.11, it follows that

$$g \circ f = \mathsf{id}_A \circ (g \circ f) \sim (\lambda(x : A).a) \circ (g \circ f) \equiv \lambda(x : A).a \sim \mathsf{id}_A.$$

Analogously, one proves $f \circ g \sim \mathsf{id}_B$. Thus, f is an equivalence. □

Lemma 4.5. *Let $P : A \to \mathcal{U}$. If A is contractible with center a, then $\left(\sum_{(x:A)} P(x)\right) \simeq P(a)$.*

Proof. Since a is the center of contraction of A, there is a function $c : \prod_{(x:A)}(a = x)$. Let T denote $\sum_{(x:A)} P(x)$. We define $f : T \to P(a)$ and $g : P(a) \to T$ by

$$f :\equiv \lambda(w : T).c(\mathsf{pr}_1(w))^{-1}{}_*(\mathsf{pr}_2(w)),$$
$$g :\equiv \lambda(z : P(a)).(a, z).$$

By Lemma 4.2, A is a set. So, there is an equality of paths $q : c(a)^{-1} = \mathsf{refl}_a^{-1}$. For each $z : P(a)$, consider the function $e_z :\equiv \lambda(p : a = a).p_*(z)$. By $\mathsf{ap}_{e_z}(q)$, we have:

$$f(g(z)) \equiv f((a, z)) \equiv c(a)^{-1}{}_*(z) = (\mathsf{refl}_a^{-1})_*(z) \equiv (\mathsf{refl}_a)_*(z) \equiv z.$$

On the other hand, for all $w : T$,

$$g(f(w)) \equiv g(c(\mathsf{pr}_1(w))^{-1}{}_*(\mathsf{pr}_2(w))) \equiv (a, c(\mathsf{pr}_1(w))^{-1}{}_*(\mathsf{pr}_2(w))),$$

so that

$$\begin{aligned}(g(f(w)) = w) &\equiv \Big(\big(a, c(\mathsf{pr}_1(w))^{-1}{}_*(\mathsf{pr}_2(w))\big) = (\mathsf{pr}_1(w), \mathsf{pr}_2(w))\Big) \\ &\simeq \sum_{p:a=\mathsf{pr}_1(w)} p_*\left(c(\mathsf{pr}_1(w))^{-1}{}_*(\mathsf{pr}_2(w))\right) = \mathsf{pr}_2(w).\end{aligned}$$

From this, since $c(\mathsf{pr}_1(w)) : a = \mathsf{pr}_1(w)$ and

$$\mathsf{ittranspinv}_2^P(c(\mathsf{pr}_1(w)), \mathsf{pr}_2(w)) : c(\mathsf{pr}_1(w))_*\left(c(\mathsf{pr}_1(w))^{-1}{}_*(\mathsf{pr}_2(w))\right) = \mathsf{pr}_2(w),$$

one obtains an element of $g(f(w)) = w$. □

4.2 Fiberwise equivalences

Definition 4.2. The fiber of a map $f : A \to B$ over a point $y : B$ is the type

$$\mathsf{fib}_f(y) :\equiv \sum_{x:A}(f(x) = y).$$

We usually think of $\mathsf{fib}_f(y)$ as the inverse image of y under f. So, we sometimes consider an element of $\mathsf{fib}_f(y)$ as being its first coordinate.

Lemma 4.6. *For all $f : A \to B$, $y : B$, and $(x, p), (x', p') : \mathsf{fib}_f(y)$,*

$$((x, p) = (x', p')) \simeq \left(\sum_{\gamma:x=x'} \mathsf{ap}_f(\gamma) \centerdot p' = p\right).$$

Proof. Let P denote the type family over B defined by $P(u) :\equiv (u = y)$. We have:

$$\begin{aligned}((x, p) = (x', p')) &\simeq \sum_{\gamma:x=x'} \mathsf{transp}^{P\circ f}(\gamma, p) = p' && \text{by Theorem 3.5} \\ &\simeq \sum_{\gamma:x=x'} \mathsf{transp}^{P}(\mathsf{ap}_f(\gamma), p) = p' && \text{by Lemmas 1.9 and 2.7(b)} \\ &\simeq \sum_{\gamma:x=x'} \mathsf{ap}_f(\gamma)^{-1} \centerdot q = p'. && \text{by Lemmas 2.7(b) and 3.2(b)}\end{aligned}$$

Now, observe that by concatenating both sides of $\mathsf{ap}_f(\gamma)^{-1} \centerdot q = p'$ with $\mathsf{ap}_f(\gamma)$ to the left, we obtain an equivalent type $q = \mathsf{ap}_f(\gamma) \centerdot p'$, which in turn is equivalent to $\mathsf{ap}_f(\gamma) \centerdot p' = q$, since path inversion is an equivalence (by Example 2.1). Therefore, by Lemma 2.7(b),

$$\left(\sum_{\gamma : x = x'} \mathsf{ap}_f(\gamma)^{-1} \centerdot q = p'\right) \simeq \left(\sum_{\gamma : x = x'} \mathsf{ap}_f(\gamma) \centerdot p' = q\right).$$

□

Definition 4.3. Let P and Q be type families over A. A *fiberwise transformation* or a *fiberwise map* between P and Q is a function $f : \prod_{(x:A)} P(x) \to Q(x)$. A fiberwise transformation is said to be a *fiberwise equivalence* if, for all $x : A$, the function $f(x) : P(x) \to Q(x)$ is an equivalence.

By the recursion principle for Σ-types, each fiberwise transformation f between type families $P : A \to \mathcal{U}$ and $Q : A \to \mathcal{U}$ induces a function

$$\mathsf{total}(f) : \left(\sum_{x:A} P(x)\right) \to \sum_{x:A} Q(x) \tag{4.1}$$

satisfying $\mathsf{total}(f)((x, u)) \equiv (x, f(x)(u))$ for each $x : A$ and $u : P(x)$.

Definition 4.4. A function $f : A \to B$ is contractible if every fiber of f is contractible. Equivalently, f is contractible if the type

$$\mathsf{isContr}(f) :\equiv \prod_{b:B} \mathsf{iscontr}(\mathsf{fib}_f(b))$$

is inhabited.

Theorem 4.7. *Every contractible map $f : A \to B$ is an equivalence.*

Proof. Let P be the element of $\mathsf{isContr}(f)$ given by hypothesis. Recall that

$$\mathsf{isContr}(f) \equiv \prod_{(y:B)} \sum_{(a:\mathsf{fib}_f(y))} \prod_{(u:\mathsf{fib}_f(y))} a = u.$$

The most natural choice of a candidate $g : B \to A$ for quasi-inverse of f consists of sending each $y : B$ to the center of contraction $\mathsf{pr}_1(P(y))$ of $\mathsf{fib}_f(y)$, or more precisely, to the element of A that appears in $\mathsf{pr}_1(P(y))$ as one of its coordinates:

$$g :\equiv \lambda(y : B).\mathsf{pr}_1(\mathsf{pr}_1(P(y))).$$

By definition of $\mathsf{isContr}(f)$, the second coordinate of $\mathsf{pr}_1(P(y))$ is an element of the type $f(\mathsf{pr}_1(\mathsf{pr}_1(P(y)))) = y$, i.e.,

$$\mathsf{pr}_2(\mathsf{pr}_1(P(y))) : f(g(y)) = y,$$

for each $y : B$. So, we define

$$h :\equiv \lambda(y : B).\mathsf{pr}_2(\mathsf{pr}_1(P(y))) : f \circ g \sim \mathsf{id}_B.$$

It remains to construct a homotopy $k : g \circ f \sim \mathsf{id}_A$. A key idea for this final step is to notice that both $g(f(x))$ and x "are in" $\mathsf{fib}_f(f(x))$, in the sense that both $f(g(f(x)))$ and $f(x)$ are equal to $f(x)$. So, since $\mathsf{fib}_f(f(x))$ is contractible, there must be a path between $g(f(x))$ and x. In fact, from $h(f(x)) : f(g(f(x))) = f(x)$ and $\mathsf{refl}_{f(x)} : f(x) = f(x)$, it follows that

$$\begin{aligned}(g(f(x)), h(f(x))) &: \mathsf{fib}_f(f(x)),\\ (x, \mathsf{refl}_{f(x)}) &: \mathsf{fib}_f(f(x)),\end{aligned}$$

and hence

$$\begin{aligned}\mathsf{pr}_2(P(f(x)))((g(f(x)), h(f(x)))) &: \mathsf{pr}_1(P(f(x))) = (g(f(x)), h(f(x))),\\ \mathsf{pr}_2(P(f(x)))((x, \mathsf{refl}_{f(x)})) &: \mathsf{pr}_1(P(f(x))) = (x, \mathsf{refl}_{f(x)}),\end{aligned}$$

so that

$$\alpha(x) :\equiv (\mathsf{pr}_2(P(f(x)))((g(f(x)), h(f(x)))))^{-1} \centerdot \mathsf{pr}_2(P(f(x)))((x, \mathsf{refl}_{f(x)}))$$

is a path between $(g(f(x)), h(f(x)))$ and $(x, \mathsf{refl}_{f(x)})$ in $\mathsf{fib}_f(f(x))$. Then, by Lemma 4.6, we get a path $k(x) : g(f(x)) = x$, defined as the first coordinate of the result of passing $\alpha(x)$ across the equivalence of that lemma. □

Theorem 4.8. *If $f : A \to B$ is an equivalence, then f is contractible.*

Proof. Let $(g, (h, k))$ be the quasi-inverse of f given by hypothesis. Fix $y : B$. Since $h(y) : f(g(y)) = y$, we have $(g(y), h(y)) : \mathsf{fib}_f(y)$. To show that $(g(y), h(y))$ is the center of contraction of $\mathsf{fib}_f(y)$, we must construct an element of the type

$$\prod_{u:\mathsf{fib}_f(y)} (g(y), h(y)) = u,$$

which is equivalent to

$$\prod_{(u:\mathsf{fib}_f(y))} \sum_{(\gamma:g(y)=\mathsf{pr}_1(u))} \mathsf{ap}_f(\gamma) \centerdot \mathsf{pr}_2(u) = h(y), \tag{4.2}$$

by Lemmas 4.6 and 2.7(a). For each $u : \mathsf{fib}_f(y)$, we have that $\mathsf{pr}_2(u) : f(\mathsf{pr}_1(u)) = y$, whence $\mathsf{ap}_g(\mathsf{pr}_2(u)) : g(f(\mathsf{pr}_1(u))) = g(y)$. From this, by $k(\mathsf{pr}_1(u)) : g(f(\mathsf{pr}_1(u))) = \mathsf{pr}_1(u)$, we have:

$$\gamma_u :\equiv \mathsf{ap}_g(\mathsf{pr}_2(u))^{-1} \centerdot k(\mathsf{pr}_1(u)) : g(y) = \mathsf{pr}_1(u).$$

To find an inhabitant of (4.2), it remains to give a path from $\mathsf{ap}_f(\gamma_u) \centerdot \mathsf{pr}_2(u)$ to $h(y)$, for any $u : \mathsf{fib}_f(y)$. By Lemma 2.5, we may suppose without loss of generality that we have a path

$$q : \mathsf{ap}_f(k(\mathsf{pr}_1(u))) = h(f(\mathsf{pr}_1(u))).$$

(If it is not the case that $\mathsf{ap}_f(k(\mathsf{pr}_1(u))) = h(f(\mathsf{pr}_1(u)))$, then replace h with the h' given by Lemma 2.5.) And we also have paths

$$\begin{aligned}
&r : \mathsf{ap}_f(\gamma_u) = \mathsf{ap}_{f\circ g}(\mathsf{pr}_2(u))^{-1} \centerdot \mathsf{ap}_f(k(\mathsf{pr}_1(u))), && \text{by Lemma 1.4}\\
&s : h(f(\mathsf{pr}_1(u))) \centerdot \mathsf{pr}_2(u) = \mathsf{ap}_{f\circ g}(\mathsf{pr}_2(u)) \centerdot h(y), && \text{by Lemma 1.12}\\
&t : \mathsf{ap}_{f\circ g}(\mathsf{pr}_2(u))^{-1} \centerdot \mathsf{ap}_{f\circ g}(\mathsf{pr}_2(u)) \centerdot h(y) = h(y). && \text{by Lemma 1.2}
\end{aligned}$$

So,

$$\begin{aligned}
\mathsf{ap}_f(\gamma_u) \centerdot \mathsf{pr}_2(u) &= (\mathsf{ap}_{f\circ g}(\mathsf{pr}_2(u))^{-1} \centerdot \mathsf{ap}_f(k(\mathsf{pr}_1(u)))) \centerdot \mathsf{pr}_2(u) && \text{by } \mathsf{ap}_\mu(r)\\
&= (\mathsf{ap}_{f\circ g}(\mathsf{pr}_2(u))^{-1} \centerdot h(f(\mathsf{pr}_1(u)))) \centerdot \mathsf{pr}_2(u) && \text{by } \mathsf{ap}_\nu(q)\\
&= \mathsf{ap}_{f\circ g}(\mathsf{pr}_2(u))^{-1} \centerdot (h(f(\mathsf{pr}_1(u))) \centerdot \mathsf{pr}_2(u)) && \text{by (Lemma 1.2(f))}^{-1}\\
&= \mathsf{ap}_{f\circ g}(\mathsf{pr}_2(u))^{-1} \centerdot (\mathsf{ap}_{f\circ g}(\mathsf{pr}_2(u)) \centerdot h(y)) && \text{by } \mathsf{ap}_\xi(s)\\
&= (\mathsf{ap}_{f\circ g}(\mathsf{pr}_2(u))^{-1} \centerdot \mathsf{ap}_{f\circ g}(\mathsf{pr}_2(u))) \centerdot h(y) && \text{by Lemma 1.2(f)}\\
&= h(y), && \text{by } t
\end{aligned}$$

where μ, ν, and ξ are functions defined by

$$\begin{aligned}
\mu(p) &:\equiv p \centerdot \mathsf{pr}_2(u),\\
\nu(p) &:\equiv (\mathsf{ap}_{f\circ g}(\mathsf{pr}_2(u))^{-1} \centerdot p) \centerdot \mathsf{pr}_2(u),\\
\xi(p) &:\equiv \mathsf{ap}_{f\circ g}(\mathsf{pr}_2(u))^{-1} \centerdot p.
\end{aligned}$$

□

Theorem 4.9. *Let f be a fiberwise transformation between type families P and Q over A. For all $a : A$ and $v : Q(a)$,*

$$\mathsf{fib}_{\mathsf{total}(f)}((a, v)) \simeq \mathsf{fib}_{f(a)}(v).$$

Proof. Consider the type family $R : \left(\sum_{(x:A)}(x = a)\right) \to \mathcal{U}$ given by

$$R(w) :\equiv \sum_{u:P(\mathsf{pr}_1(w))} (\mathsf{pr}_2(w))_* (f(\mathsf{pr}_1(w), u)) = v.$$

We have:

$$\begin{aligned}
\mathsf{fib}_{\mathsf{total}(f)}((a,v)) &\equiv \left(\sum_{w:\sum_{(x:A)} P(x)} (\mathsf{pr}_1(w), f(\mathsf{pr}_1(w), \mathsf{pr}_2(w))) = (a,v)\right) \\
&\simeq \sum_{(x:A)} \sum_{(u:P(x))} (x, f(x,u)) = (a,v) && \text{by (Lemma 2.3)}^{-1} \\
&\simeq \sum_{(x:A)} \sum_{(u:P(x))} \sum_{(p:x=a)} p_*(f(x,u)) = v && \text{by Theorem 3.5 and Lemma 2.7(b)} \\
&\simeq \sum_{(x:A)} \sum_{(p:x=a)} R((x,p)) && \text{by Lemmas 2.4 and 2.7(b)} \\
&\simeq \sum_{w:\sum_{(x:A)}(x=a)} R(w) && \text{by Lemma 2.3} \\
&\simeq \sum_{u:P(a)} (\mathsf{refl}_a)_*(f(a,u)) = v. && \text{by Lemmas 4.3(b) and 4.5}
\end{aligned}$$

Since this last type is definitionally equal to $\mathsf{fib}_{f(a)}(v)$, the theorem is proved. □

The following theorem is a version of a well-known result in algebraic topology which can be found in (MAY, 1999, Section 7.5, p. 52).

Theorem 4.10. *Let A be a type and f a fiberwise transformation between families P and Q over A. If $\mathsf{total}(f)$ is an equivalence, then f is a fiberwise equivalence.*

Proof. First, note that there is a function of type

$$\left(\prod_{w:\sum_{(x:A)} Q(x)} \mathsf{iscontr}(\mathsf{fib}_{\mathsf{total}(f)}(w))\right) \to \left(\prod_{(a:A)} \prod_{(v:Q(a))} \mathsf{iscontr}(\mathsf{fib}_{\mathsf{total}(f)}((a,v)))\right) \tag{4.3}$$

that maps each g in its domain to $\lambda(a:A).\lambda(v:Q(a)).g((a,v))$. Now, consider the composite function

$$\begin{aligned}
&\mathsf{qinv}(\mathsf{total}(f)) \\
&\quad\to \mathsf{isContr}(\mathsf{total}(f)) && \text{by Theorem 4.8} \\
&\quad\equiv \prod_{w:\sum_{(x:A)} Q(x)} \mathsf{iscontr}(\mathsf{fib}_{\mathsf{total}(f)}(w)) \\
&\quad\to \prod_{(a:A)} \prod_{(v:Q(a))} \mathsf{iscontr}(\mathsf{fib}_{\mathsf{total}(f)}((a,v))) && \text{by (4.3)} \\
&\quad\to \prod_{(a:A)} \prod_{(v:Q(a))} \mathsf{iscontr}(\mathsf{fib}_{f(a)}(v)) && \text{by Theorem 4.9 and Lemma 2.7(a)} \\
&\quad\equiv \prod_{a:A} \mathsf{isContr}(f(a)) \\
&\quad\to \prod_{a:A} \mathsf{qinv}(f(a)). && \text{by Theorem 4.7 and Lemma 2.7(a)}
\end{aligned}$$

□

5 The Fundamental Group of the Circle

The goal of this chapter is to reproduce a proof of $\pi_1(\mathbb{S}^1) = \mathbb{Z}$ in type theory. This was originally done by Michael Shulman and posteriorly (in another approach) by Daniel Licata (LICATA; SHULMAN, 2013). Shulman have translated into type theory a standard proof of $\pi_1(\mathbb{S}^1) = \mathbb{Z}$ using covering spaces (HATCHER, 2002, Chapter 1). In the sequel, we aim to explore this translation.

5.1 Inductive definitions

Just like free groups can be defined by a set of symbols (its generators), types can be defined by a list of *constructors*. A constructor of a type X is a symbol denoting a function of some number of arguments (possibly zero) with codomain X and domains being types. A constructor of X with zero arguments is an element of X, considered as a constant function.

This pattern of type definition is known as *induction*. A type defined by a list of constructors is said to be an *inductive type*. Once a type X have been inductively defined by n constructors

$$c_i : A_{i,1} \to \cdots \to A_{i,k_i} \to X \quad (i \in \{1, \ldots, n\}),$$

one can define any function $f : \prod_{(x:X)} P(x)$ by assigning a unique element $f(c_i(a_1, \ldots, a_{k_i}))$ of $P(c_i(a_1, \ldots, a_{k_i}))$ to each expression $c_i(a_1, \ldots, a_{k_i})$, with $a_j : A_{i,j}$, $j \in \{1, \ldots, k_i\}$, and $i \in \{1, \ldots, n\}$.

Example 5.1. *The type of natural numbers* $\mathbb{N}$ *is generated by*

- $0_{\mathbb{N}} : \mathbb{N}$
- $\mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$.

Example 5.2. *The type of booleans* $\mathbf{2}$ *is inductively defined by*

- $0_{\mathbf{2}} : \mathbf{2}$
- $1_{\mathbf{2}} : \mathbf{2}$.

This type corresponds to the set of boolean values $\{0, 1\}$*, which contains only two elements.*

A new feature of the type theory currently used in homotopy type theory is that it admits a broader notion of inductive definition, known as *higher inductive definition*. In

this new pattern, the constructors may originate not only elements of X but also paths in X. For instance, there may be constructors of the form

$$c : A_1 \to \cdots \to A_k \to (b(a_1) =_X b(a_2)),$$

where $b : A \to X$ is another constructor of X and $a_1, a_2 : A$. Given a higher inductive type X, any function $f : X \to Y$ may be defined by assigning a unique element of Y (respectively, a unique element of Y) to each element (respectively, path) originated from some constructor of X. The assignments of paths must be coherent with the assignments of the endpoints of the paths. For instance, if $c_1 : A \to X$ and $c_2 : A \to (c_1(a_1) = c_1(a_2))$, with $a_1, a_2 : A$, are constructors of X and we assign $y_1 : Y$ and $y_2 : Y$ to $c_1(a_1)$ and $c_1(a_2)$, respectively, then the path in Y to be assigned to $c_2(a)$, for any $a : A$, must have type $y_1 =_Y y_2$. Similar considerations apply for dependent functions $f : \prod_{(x:X)} P(x)$ and dependent constructors, with dependent paths wherever necessary (see (Univalent Foundations Program, 2013, Section 6.2)).

Example 5.3. *The circle $\mathbb{S}^1$ is defined as being generated by the following constructors:*

- $\mathsf{base} : \mathbb{S}^1$
- $\mathsf{loop} : \mathsf{base} =_{\mathbb{S}^1} \mathsf{base}$.

The goal of this chapter is to prove a well-known homotopical property of the circle, namely that its fundamental group is isomorphic to the additive group of integers. We refer the reader to (Univalent Foundations Program, 2013, Section 6.10) for a definition of the type of integers $\mathbb{Z}$. We assume that $\mathbb{Z}$ is endowed with an element $0 : \mathbb{Z}$, functions

- $\mathsf{i} : \mathbb{N} \to \mathbb{Z}$, written $n \mapsto n$,
- $\mathsf{succ} : \mathbb{Z} \to \mathbb{Z}$, written $n \mapsto n + 1$,
- $\mathsf{minus} : \mathbb{Z} \to \mathbb{Z}$, written $n \mapsto -n$,

and a binary relation $<$ satisfying the usual order properties of the natural and integer numbers. To be careful about the precise meaning of "binary relation," we would need to talk about *mere propositions*, but we want to avoid this. It is worth mentioning that $0 < m$ and $m < 0$ mean "$m = n + 1$ for some $n : \mathbb{N}$" and "$m = -(n + 1)$ for some $n : \mathbb{N}$," respectively. We assume that $\mathbb{Z}$ is a set (in the sense of Definition 4.1) and the function succ has a quasi-inverse $n \mapsto n - 1$. We denote by succeq the respective equivalence from $\mathbb{Z}$ to $\mathbb{Z}$, so that, for any $n : \mathbb{Z}$,

$$\mathsf{pr}_1(\mathsf{succeq}, n) \equiv \mathsf{succ}(n) \equiv n + 1. \tag{5.1}$$

Another important fact about $\mathbb{Z}$ is the following lemma, which can be found in (Univalent Foundations Program, 2013, Lemma 6.10.12).

Lemma 5.1 (Induction principle for $\mathbb{Z}$)**.** *For any $P : \mathbb{Z} \to \mathcal{U}$, there is a map $\mathsf{ind}_{\mathbb{Z}}(P)$ that assigns a function*

$$f :\equiv \mathsf{ind}_{\mathbb{Z}}(P, d_0, d_+, d_-) : \prod_{n:\mathbb{Z}} P(n)$$

to each

$$d_0 : P(0), \quad d_+ : \prod_{n:\mathbb{N}} P(n) \to P(n+1), \quad \textit{and} \quad d_- : \prod_{n:\mathbb{N}} P(-n) \to P(-(n+1))$$

in such a way that

$$f(0) \equiv d_0, \quad f(n+1) \equiv d_+(f(n)), \quad \textit{and} \quad f(-(n+1)) \equiv d_-(f(-n)),$$

for all $n : \mathbb{N}$.

5.2 The flattening lemma for the universal cover of $\mathbb{S}^1$

Definition 5.1 (Universal cover of $\mathbb{S}^1$)**.** We define $\mathsf{code} : \mathbb{S}^1 \to \mathcal{U}$ by

$$\mathsf{code}(\mathsf{base}) :\equiv \mathbb{Z},$$
$$\mathsf{ap}_{\mathsf{code}}(\mathsf{loop}) :\equiv \mathsf{ua}(\mathsf{succeq}).$$

Definition 5.2. The *homotopical reals* are the elements of the type R generated by the following constructors:

- $\mathsf{c} : \mathbb{Z} \to \mathsf{R}$
- $\mathsf{d} : \prod_{(n:\mathbb{Z})} (\mathsf{c}(n) =_{\mathsf{R}} \mathsf{c}(n+1))$.

Lemma 5.2 (Flattening Lemma for code)**.** $\sum_{(x:\mathbb{S}^1)} \mathsf{code}(x) \simeq \mathsf{R}$.

Proof. To simplify the notation, let $T :\equiv \sum_{(x:\mathbb{S}^1)} \mathsf{code}(x)$. We want to construct functions $h : \mathsf{R} \to T$, $k : T \to \mathsf{R}$, and homotopies $\alpha : h \circ k \sim \mathsf{id}_T$, $\beta : k \circ h \sim \mathsf{id}_{\mathsf{R}}$. For clarity, we use the concise notation for transport (with subscript asterisk) exclusively with respect to code.

The construction of $h : \mathsf{R} \to T$. The inductive definition of R says that in order to give (intuitively) a value $h(r)$ for each $r : \mathsf{R}$, it suffices to evaluate h on the elements of the form $\mathsf{c}(n)$, with $n : \mathbb{Z}$, and on the paths $\mathsf{d}(n) : \mathsf{c}(n) =_T \mathsf{c}(n+1)$ between these. More precisely, we must give:

- for each $n : \mathbb{Z}$, an element $h(\mathsf{c}(n))$ of T;
- for each $n : \mathbb{Z}$, a path $\mathsf{ap}_h(\mathsf{d}(n)) : (h(\mathsf{c}(n)) =_T h(\mathsf{c}(n+1)))$.

For the first coordinate of $h(\mathsf{c}(n))$, we do not have much choice: the only distinguished element of $\mathbb{S}^1$ is base. For the second one, we may take n. So,

$$h(\mathsf{c}(n)) :\equiv (\mathsf{base}, n). \tag{5.2}$$

Now, we use the characterization of paths in dependent pair types (Theorem 3.5):

$$\left(\sum_{p:\mathsf{base}=\mathsf{base}} \mathsf{transp}^{\mathsf{code}}(p, n) =_{\mathbb{Z}} n+1\right) \simeq ((\mathsf{base}, n) =_T (\mathsf{base}, n+1)).$$

Denoting by p_n the composite path

$$\begin{aligned}
\mathsf{transp}^{\mathsf{code}}(\mathsf{loop}, n) &\equiv \mathsf{transp}^{(X\mapsto X)\circ \mathsf{code}}(\mathsf{loop}, n) \\
&= \mathsf{transp}^{(X\mapsto X)}(\mathsf{ap}_{\mathsf{code}}(\mathsf{loop}), n) && \text{by Lemma 1.9} \\
&\equiv \mathsf{transp}^{(X\mapsto X)}(\mathsf{ua}(\mathsf{succeq}), n) \\
&= \mathsf{succ}(n) \equiv n+1, && \text{by (2.8) and (5.1)}
\end{aligned}$$

we have that $\mathsf{dpair}^{=}(\mathsf{loop}, p_n)$ has type $h(\mathsf{c}(n)) =_T h(\mathsf{c}(n+1))$. So, we define:

$$\mathsf{ap}_h(\mathsf{d}(n)) :\equiv \mathsf{dpair}^{=}(\mathsf{loop}, p_n). \tag{5.3}$$

The construction of $k : T \to \mathsf{R}$. We want to define k using the recursion principle for dependent pair types. To do this, we need an equivalence

$$\eta : \left(\prod_{n:\mathbb{Z}} \mathsf{c}(n) =_{\mathsf{R}} \mathsf{c}(n+1)\right) \simeq \left(\mathsf{c} =^{Q}_{\mathsf{loop}} \mathsf{c}\right),$$

where $Q : \mathbb{S}^1 \to \mathcal{U}$ is given by $Q(x) :\equiv (\mathsf{code}(x) \to \mathsf{R})$. For each $n : \mathbb{Z}$,

$$\mathsf{ap}_{\mathsf{c}}(p_n) : \mathsf{c}(\mathsf{loop}_*(n)) =_{\mathsf{R}} \mathsf{c}(n+1), \tag{5.4}$$

$$\mathsf{transpconst}^{\mathsf{R}}_{\mathsf{loop}}(\mathsf{c}(n)) : \mathsf{transp}^{x\mapsto \mathsf{R}}(\mathsf{loop}, \mathsf{c}(n)) =_{\mathsf{R}} \mathsf{c}(n), \tag{5.5}$$

where the symbol $x \mapsto \mathsf{R}$ denotes the constant type family defined on $\mathbb{S}^1$ that gives the value R for any input. From this, by Lemma 2.2, we have an equivalence

$$\varepsilon_n : (\mathsf{c}(n) =_{\mathsf{R}} \mathsf{c}(n+1)) \simeq \left(\mathsf{transp}^{x\mapsto \mathsf{R}}(\mathsf{loop}, \mathsf{c}(n)) =_{\mathsf{R}} \mathsf{c}(\mathsf{loop}_*(n))\right) \tag{5.6}$$

such that

$$\mathsf{pr}_1(\varepsilon_n) :\equiv \lambda(p : \mathsf{c}(n) =_{\mathsf{R}} \mathsf{c}(n+1)).\left(\mathsf{transpconst}^{\mathsf{R}}_{\mathsf{loop}}(\mathsf{c}(n)) \centerdot p \centerdot \mathsf{ap}_{\mathsf{c}}(p_n)^{-1}\right), \tag{5.7}$$

for each $n : \mathbb{Z}$. These equivalences produce, by Lemma 2.7(a), another equivalence

$$\varepsilon : \left(\prod_{n:\mathbb{Z}} \mathsf{c}(n) =_{\mathsf{R}} \mathsf{c}(n+1)\right) \simeq \left(\prod_{n:\mathbb{Z}} \mathsf{transp}^{x\mapsto \mathsf{R}}(\mathsf{loop}, \mathsf{c}(n)) =_{\mathsf{R}} \mathsf{c}(\mathsf{loop}_*(n))\right)$$

such that

$$\mathsf{pr}_1(\varepsilon, f, n) :\equiv \mathsf{pr}_1(\varepsilon_n, f(n)) \tag{5.8}$$

for any $f : \prod_{(n:\mathbb{Z})} \mathsf{c}(n) =_{\mathsf{R}} \mathsf{c}(n+1)$ and $n : \mathbb{Z}$. Moreover, by Lemma 3.9, we have an equivalence

$$\varphi : \left(\mathsf{transp}^{Q}(\mathsf{loop}, \mathsf{c}) =_{\mathbb{Z}\to\mathsf{R}} \mathsf{c}\right) \simeq \left(\prod_{n:\mathbb{Z}} \mathsf{transp}^{x\mapsto\mathsf{R}}(\mathsf{loop}, \mathsf{c}(n)) =_{\mathsf{R}} \mathsf{c}(\mathsf{loop}_*(n))\right). \tag{5.9}$$

So, since $(\mathsf{c} =^{Q}_{\mathsf{loop}} \mathsf{c}) \equiv \left(\mathsf{transp}^{Q}(\mathsf{loop}, \mathsf{c}) =_{\mathbb{Z}\to\mathsf{R}} \mathsf{c}\right)$, we may define

$$\eta :\equiv \varphi^{-1} \circ \varepsilon. \tag{5.10}$$

Now, let $g : \prod_{(x:\mathbb{S}^1)} Q(x)$ be given by $\mathbb{S}^1$-induction as follows:

$$g(\mathsf{base}) :\equiv \mathsf{c}, \tag{5.11}$$

$$\mathsf{apd}_g(\mathsf{loop}) :\equiv \mathsf{pr}_1(\eta, \mathsf{d}). \tag{5.12}$$

We define k from g by the recursion principle for T so that

$$k((x, n)) :\equiv g(x)(n),$$

for any $x : \mathbb{S}^1$ and $n : \mathbb{Z}$. That is, $k :\equiv \mathsf{rec}_T(\mathsf{R}, g)$.

The construction of $\alpha : h \circ k \sim \mathsf{id}_T$. To construct α by induction on T, it suffices to give a function

$$j : \prod_{(x:\mathbb{S}^1)} \prod_{(u:\mathsf{code}(x))} h(k((x, u))) = (x, u).$$

Using induction on $\mathbb{S}^1$, this can be done by giving

- a function $j(\mathsf{base}) : \prod_{(n:\mathbb{Z})} h(k((\mathsf{base}, n))) = (\mathsf{base}, n)$;
- a dependent path $\mathsf{apd}_j(p) : j(\mathsf{base}) =^{\Pi_A(B)}_{\mathsf{loop}} j(\mathsf{base})$, where $\Pi_A(B)$ is the type family defined as in (3.7) for $A :\equiv \mathsf{code}$ and $B :\equiv \lambda x.\lambda u.\, (h(k((x, u))) = (x, u))$.

For all $n : \mathbb{Z}$, we have:

$$h(k((\mathsf{base}, n))) \equiv h(g(\mathsf{base})(n)) \equiv h(\mathsf{c}(n)) \equiv (\mathsf{base}, n).$$

Then, we may define:

$$j(\mathsf{base}) :\equiv \lambda(n : \mathbb{Z}).\mathsf{refl}_{(\mathsf{base},n)}.$$

Now, by Lemma 3.11,

$$\begin{aligned}
&\left(j(\mathsf{base}) =^{\Pi_A(B)}_{\mathsf{loop}} j(\mathsf{base})\right) \\
&\quad\equiv \left(\mathsf{transp}^{\Pi_A(B)}(\mathsf{loop}, j(\mathsf{base})) = j(\mathsf{base})\right) \\
&\quad\simeq \prod_{n:\mathbb{Z}} \left(\mathsf{transp}^{\widehat{B}}\left(\mathsf{dpair}^{=}\left(\mathsf{loop},\ \mathsf{refl}_{\mathsf{loop}_*(n)}\right),\ j(\mathsf{base}, n)\right) = j(\mathsf{base}, \mathsf{loop}_*(n))\right),
\end{aligned}$$

where $\widehat{B} : \left(\sum_{(x:\mathbb{S}^1)} \mathsf{code}(x)\right) \to \mathcal{U}$ is defined as in (3.8). So, it remains to prove that

$$\mathsf{transp}^{\widehat{B}}\left(\mathsf{dpair}^{=}\left(\mathsf{loop},\ \mathsf{refl}_{\mathsf{loop}_*(n)}\right),\ j(\mathsf{base}, n)\right) = j(\mathsf{base}, \mathsf{loop}_*(n)), \tag{5.13}$$

for any $n : \mathbb{Z}$. Let $\ell :\equiv \lambda(m : \mathbb{Z}).(\mathsf{base}, m)$. Thus, $j(\mathsf{base}, m) : \widehat{B}(\ell(m))$ for all $m : \mathbb{Z}$. Recall that $p_n : \mathsf{loop}_*(n) = n + 1$. Since

$$\begin{aligned} j(\mathsf{base}, \mathsf{loop}_*(n)) &= \mathsf{transp}^{\widehat{B}\circ\ell}(p_n^{-1}, j(\mathsf{base}, n+1)) && \text{by } \left(\mathsf{apd}_{j(\mathsf{base})}(p_n^{-1})\right)^{-1} \\ &= \mathsf{transp}^{\widehat{B}}(\mathsf{ap}_\ell(p_n^{-1}), j(\mathsf{base}, n+1)) && \text{by Lemma 1.9} \\ &= \mathsf{transp}^{\widehat{B}}(\mathsf{ap}_\ell(p_n)^{-1}, j(\mathsf{base}, n+1)), && \text{by Lemma 1.4(b)} \end{aligned}$$

we have (by Lemma (2.2)) that the type (5.13) is equivalent to

$$\begin{aligned} \mathsf{transp}^{\widehat{B}}\left(\mathsf{dpair}^{=}\left(\mathsf{loop},\ \mathsf{refl}_{\mathsf{loop}_*(n)}\right),\ j(\mathsf{base}, n)\right) \\ = \mathsf{transp}^{\widehat{B}}(\mathsf{ap}_\ell(p_n)^{-1}, j(\mathsf{base}, n+1)). \end{aligned} \tag{5.14}$$

By applying $\lambda b.\mathsf{transp}^{\widehat{B}}(\mathsf{ap}_\ell(p_n), b)$ on both sides of (5.14), we obtain, by Lemmas 1.3, 1.7, and Corollary 1.8, another identity type

$$\mathsf{transp}^{\widehat{B}}\left(\mathsf{ap}_\ell(p_n) \centerdot \mathsf{dpair}^{=}(\mathsf{loop},\ \mathsf{refl}_{\mathsf{loop}_*(n)}),\ j(\mathsf{base}, n)\right) = j(\mathsf{base}, n+1) \tag{5.15}$$

which is equivalent to (5.13), since we can recover (5.14) by transporting both sides of (5.15) along $\mathsf{ap}_\ell(p_n)^{-1}$ with respect to $\widehat{B}$ (Example 2.3). But, by Lemmas 3.8 and 3.7,

$$\begin{gathered} \mathsf{ap}_\ell(p_n) = \mathsf{dpair}^{=}(\mathsf{refl}_{\mathsf{base}}, p_n), \\ \mathsf{dpair}^{=}(\mathsf{refl}_{\mathsf{base}}, p_n) \centerdot \mathsf{dpair}^{=}(\mathsf{loop},\ \mathsf{refl}_{\mathsf{loop}_*(n)}) = \mathsf{dpair}^{=}(\mathsf{loop}, p_n), \end{gathered}$$

so that

$$\begin{aligned} \mathsf{ap}_\ell(p_n) \centerdot \mathsf{dpair}^{=}(\mathsf{loop},\ \mathsf{refl}_{\mathsf{loop}_*(n)}) &= \mathsf{dpair}^{=}(\mathsf{refl}_{\mathsf{base}}, p_n) \centerdot \mathsf{dpair}^{=}(\mathsf{loop},\ \mathsf{refl}_{\mathsf{loop}_*(n)}) \\ &= \mathsf{dpair}^{=}(\mathsf{loop}, p_n). \end{aligned}$$

Thus, (5.15) is equivalent to

$$\mathsf{transp}^{\widehat{B}}\left(\mathsf{dpair}^{=}(\mathsf{loop}, p_n),\ j(\mathsf{base}, n)\right) = j(\mathsf{base}, n+1). \tag{5.16}$$

By path induction on $r : x =_{\mathbb{S}^1} y$ and based path induction on $s : \mathsf{transp}^{\mathsf{code}}(r, u) = v$, its easy to verify that

$$\mathsf{transp}^{\widehat{B}}(\mathsf{dpair}^{=}(r, s), \mathsf{refl}_{(x,u)}) = \mathsf{refl}_{(y,v)},$$

for r and s arbitrary. From this, since $j(\mathsf{base}) \equiv \lambda(m : \mathbb{Z}).\mathsf{refl}_{(\mathsf{base},m)}$, it follows that (5.16) is true.

The construction of $\beta : k \circ h \sim \mathsf{id}_{\mathsf{R}}$. To construct β by R-induction, it suffices to prove that $k(h(\mathsf{c}(n))) = \mathsf{c}(n)$ and $\mathsf{ap}_k(\mathsf{ap}_h(\mathsf{d}(n))) = \mathsf{d}(n)$, for any $n : \mathbb{Z}$. The first equality follows immediately from $k(h(\mathsf{c}(n))) \equiv k((\mathsf{base}, n)) \equiv \mathsf{c}(n)$. Now, by (5.3), it remains to show that $\mathsf{ap}_k(\mathsf{dpair}^{=}(\mathsf{loop}, p_n)) = \mathsf{d}(n)$. First, we observe that for all $r : x_1 =_{\mathbb{S}^1} x_2$ and $s : r_*(y_1) = y_2$, the path $\mathsf{ap}_k(\mathsf{dpair}^{=}(r, s)) : k((x_1, y_1)) = k((x_2, y_2))$ is equal to the composite

$$\begin{aligned}
k((x_1, y_1)) \equiv g(x_1)(y_1) &= \mathsf{transp}^{x \mapsto \mathsf{R}}(r, g(x_1)(y_1)) && \text{by } q_1(r, s)\\
&= \mathsf{transp}^{x \mapsto \mathsf{R}}(r, g(x_1)({r^{-1}}_*(r_*(y_1)))) && \text{by } q_2(r, s)\\
&= \mathsf{transp}^{Q}(r, g(x_1))(r_*(y_1)) && \text{by } q_3(r, s)\\
&= g(x_2)(r_*(y_1)) && \text{by } q_4(r, s)\\
&= g(x_2)(y_2) \equiv k((x_2, y_2)), && \text{by } q_5(r, s)
\end{aligned}$$

where

$$\begin{aligned}
q_1(r, s) &:\equiv (\mathsf{transpconst}^{\mathsf{R}}_r(g(x_1)(y_1)))^{-1},\\
q_2(r, s) &:\equiv \mathsf{ap}_{u(r,s)}((\mathsf{ittranspinv}_1^{\mathsf{code}}(r, y_1))^{-1})\\
&\qquad \text{for } u(r, s) :\equiv \mathsf{transp}^{x \mapsto \mathsf{R}}(r) \circ g(x_1),\\
q_3(r, s) &:\equiv \mathsf{ap}_{e(r,s)}((\mathsf{transpfun}_r^{\mathsf{code} \to (x \mapsto \mathsf{R})}(g(x_1)))^{-1})\\
&\qquad \text{for } e(r, s) :\equiv \lambda f.f(r_*(y_1)),\\
q_4(r, s) &:\equiv \mathsf{happly}(\mathsf{apd}_g(r))(r_*(y_1)),\\
q_5(r, s) &:\equiv \mathsf{ap}_{g(x_2)}(s).
\end{aligned}$$

In fact, it is easy to verify that $\mathsf{ap}_k(\mathsf{dpair}^{=}(r, s))$ and each $q_i(r, s)$, with $i = 1, \ldots, 5$, reduce to reflexivities when $r \equiv \mathsf{refl}_{x_1}$ and $s \equiv \mathsf{refl}_{y_1}$. From this, the claim follows immediately by path induction on r and s. In particular, for $r \equiv \mathsf{loop}$ and $s \equiv p_n$, we have:

$$\mathsf{ap}_k(\mathsf{dpair}^{=}(\mathsf{loop}, p_n)) = q_1(\mathsf{loop}, p_n) \centerdot \ldots \centerdot q_5(\mathsf{loop}, p_n). \tag{5.17}$$

Second, since $\mathsf{apd}_g(\mathsf{loop}) \equiv \mathsf{pr}_1(\eta, \mathsf{d})$ corresponds to $\mathsf{pr}_1(\varphi, \mathsf{pr}_1(\eta, \mathsf{d})) \equiv \mathsf{pr}_1(\varepsilon, \mathsf{d})$ under φ, we have, by the computation rule of Lemma 3.9 and by (5.8):

$$q_4(\mathsf{loop}, p_n) = q_3(\mathsf{loop}, p_n)^{-1} \centerdot q_2(\mathsf{loop}, p_n)^{-1} \centerdot \mathsf{pr}_1(\varepsilon_n, \mathsf{d}(n)). \tag{5.18}$$

Third, by (5.7),

$$\mathsf{pr}_1(\varepsilon_n, \mathsf{d}(n)) \equiv q_1(\mathsf{loop}, p_n)^{-1} \centerdot d(n) \centerdot q_5(\mathsf{loop}, p_n)^{-1} \tag{5.19}$$

Finally, substituting (5.19) and (5.18) in (5.17), we conclude that

$$\mathsf{ap}_k(\mathsf{dpair}^{=}(\mathsf{loop}, p_n)) = \mathsf{d}(n). \qquad \square$$

5.3 A proof of $\pi_1(\mathbb{S}^1) = \mathbb{Z}$

Remember that a set is a type without nontrivial paths. There is a map $\|-\|_0 : \mathcal{U} \to \mathcal{U}$, called 0-truncation, that assigns a set $\|B\|_0$ to each type B in such a way that $\|B\|_0 = B$ if B is a set (see (Univalent Foundations Program, 2013, Section 6.9)). Topologically, $\|B\|_0$ is B with the discrete topology. But $\|B\|_0$ still has the operations of path concatenation and path inversion, which make it a groupoid (Lemma 1.2). Furthermore, if $B \equiv (a = a)$ for some $a : A$, then $\|B\|_0$ is a group.

Definition 5.3. Given $A : \mathcal{U}$ and $a : A$, the *loop space* of A based on a is the type

$$\Omega(A, a) :\equiv (a = a)$$

and the fundamental group of A based on a is the type

$$\pi_1(A, a) :\equiv \|\Omega(A, a)\|_0.$$

Now we are ready to determine the fundamental group of the circle.

Corollary 5.3. *The type* $\sum_{(x:\mathbb{S}^1)}(\mathsf{base} = x)$ *is contractible.*

Proof. It is just a particular case of Lemma 4.3(a). □

Lemma 5.4. R *is contractible.*

Proof. We want to show that $\mathsf{c}(0)$ is the center of contraction of R. This means to construct a function $q : \prod_{(y:\mathsf{R})}(\mathsf{c}(0) = y)$. The inductive definition of R tells us that, in order to evaluate q (intuitively) on each element y of R, we only need concern ourselves with the elements of the form $\mathsf{c}(z)$, for $z : \mathbb{Z}$, and with the paths $\mathsf{d}(z) : \mathsf{c}(z) =_{\mathsf{R}} \mathsf{c}(z+1)$ between these. More precisely, we must give:

(i) for each $z : \mathbb{Z}$, an element $q(\mathsf{c}(z)) : Q(\mathsf{c}(z))$;

(ii) for each $z : \mathbb{Z}$, a path $\mathsf{apd}_q(\mathsf{d}(z)) : q(\mathsf{c}(z)) =^Q_{\mathsf{d}(z)} q(\mathsf{c}(z+1))$,

where $Q : \mathsf{R} \to \mathcal{U}$ is defined by $Q(y) :\equiv (\mathsf{c}(0) = y)$. Each of these items corresponds to a function on $\mathbb{Z}$. For the part (i), we use induction on $\mathbb{Z}$:

$$\begin{aligned} q(\mathsf{c}(0)) &:\equiv \mathsf{refl}_{\mathsf{c}(0)}; \\ q(\mathsf{c}(n+1)) &:\equiv q(\mathsf{c}(n)) \centerdot \mathsf{d}(n) && \text{if } 0 < n; \\ q(\mathsf{c}(n-1)) &:\equiv q(\mathsf{c}(n)) \centerdot \mathsf{d}(n-1)^{-1} && \text{if } n < 0. \end{aligned}$$

Verify that these definitions are well-typed. For the part (ii), we must construct an element of the type

$$(q(\mathsf{c}(z)) =^Q_{\mathsf{d}(z)} q(\mathsf{c}(z+1))) \equiv (\mathsf{transp}^Q(\mathsf{d}(z), q(\mathsf{c}(z))) = q(\mathsf{c}(z+1))), \tag{5.20}$$

which is equivalent (by Lemma 2.2) to

$$q(\mathsf{c}(z+1)) = q(\mathsf{c}(z+1)), \tag{5.21}$$

since $\mathsf{transp}^{Q}(\mathsf{d}(z), q(\mathsf{c}(z))) = q(\mathsf{c}(z))\mathsf{.d}(z) \equiv q(\mathsf{c}(z+1))$, by Lemma 3.2(a) and definition of $z \mapsto q(\mathsf{c}(z))$. So, we define $\mathsf{apd}_q(\mathsf{d}(z))$ as the element of (5.20) correspondent to $\mathsf{refl}_{q(\mathsf{c}(z+1))}$ under this equivalence. □

Corollary 5.5. *The type* $\sum_{(x:\mathbb{S}^1)} \mathsf{code}(x)$ *is contractible.*

Proof. Immediate from Lemmas 5.2, 5.4, and 4.1. □

We define

$$\mathsf{encode} : \prod_{x:\mathbb{S}^1} (\mathsf{base} = x) \to \mathsf{code}(x) \tag{5.22}$$

by $\mathsf{encode}(x, p) = \mathsf{transp}^{\mathsf{code}}(p, 0)$.

Corollary 5.6. *The map*

$$\mathsf{total}(\mathsf{encode}) : \left(\sum_{x:\mathbb{S}^1} \mathsf{base} = x\right) \to \sum_{x:\mathbb{S}^1} \mathsf{code}(x)$$

(see (4.1)) is an equivalence.

Proof. Immediate from Corollaries 5.3 and 5.5 and Lemma 4.4. □

Theorem 5.7. $\Omega(\mathbb{S}^1, \mathsf{base}) \simeq \mathbb{Z}$.

Proof. By Theorem 4.10 and Corollary 5.6, encode is a fiberwise equivalence. Therefore, $\mathsf{encode}(\mathsf{base}) : ((\mathsf{base} = \mathsf{base}) \simeq \mathsf{code}(\mathsf{base}))$, i.e., $\mathsf{encode}(\mathsf{base}) : \Omega(\mathbb{S}^1, \mathsf{base}) \simeq \mathbb{Z}$. □

Corollary 5.8. $\pi_1(\mathbb{S}^1, \mathsf{base}) = \mathbb{Z}$.

Proof. From Theorem 5.7, by the univalence axiom, we have that $\Omega(\mathbb{S}^1, \mathsf{base}) = \mathbb{Z}$. Hence, by Lemma 1.3, $\|\Omega(\mathbb{S}^1, \mathsf{base})\|_0 = \|\mathbb{Z}\|_0$. But $\pi_1(\mathbb{S}^1, \mathsf{base}) \equiv \|\Omega(\mathbb{S}^1, \mathsf{base})\|_0$, by definition. And $\|\mathbb{Z}\|_0 = \mathbb{Z}$, since $\mathbb{Z}$ is a set. Thus, $\pi_1(\mathbb{S}^1, \mathsf{base}) = \mathbb{Z}$. □

References

AWODEY, S. Homotopy type theory and univalent foundations of mathematics. <http://www.andrew.cmu.edu/user/awodey/hott/CMUslides.pdf>. 2012. Cited in page 10.

AWODEY, S.; WARREN, M. A. Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society*, v. 146, p. 45–55, 2009. Cited in page 10.

HATCHER, A. *Algebraic Topology*. [S.l.]: Cambridge University Press, 2002. Cited 2 times in page(s) 40 and 49.

HINDLEY, J. R.; SELDIN, J. P. *Lambda Calculus and Combinators, an Introduction*. New York: Cambridge University Press, 2008. Cited in page 11.

HOFMANN, M.; STREICHER, T. The groupoid interpretation of type theory. *Oxford Logic Guides*, v. 36, p. 83–111, 1998. Cited in page 10.

LICATA, D. R.; SHULMAN, M. Calculating the fundamental group of the circle in homotopy type theory. *LICS 2013: Proceedings of the Twenty-Eighth Annual ACM/IEEE Symposium on Logic in Computer Science*, 2013. Cited in page 49.

MARTIN-LÖF, P. An intuitionistic theory of types: predicative part. *Studies in Logic and the Foundations of Mathematics*, v. 80, p. 73–118, 1975. Cited in page 10.

MARTIN-LÖF, P. Intuitionistic type theory. *Studies in Proof Theory*, v. 1, 1984. Cited in page 10.

MARTIN-LÖF, P. An intuitionistic theory of types. *Oxford Logic Guides*, v. 36, p. 127–172, 1998. Cited in page 10.

MAY, J. P. *A Concise Course in Algebraic Topology*. [S.l.]: University Of Chicago Press, 1999. Cited in page 48.

Univalent Foundations Program, T. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: <http://homotopytypetheory.org/book>, 2013. Cited 8 times in page(s) 10, 11, 12, 13, 14, 26, 50, and 56.

VOEVODSKY, V. A very short note on the homotopy λ-calculus. <http://www.math.ias.edu/~Vladimir/Site3/Univalent_Foundations_files/Hlambda_short_current.pdf>. 2006. Cited in page 10.

VOEVODSKY, V. Univalent foundations project. <http://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/univalent_foundations_project.pdf>. 2010. Cited in page 5.