South American Journal of Logic
Vol. 4, n. 2, pp. 441–484, 2018
ISSN: 2446-6719

Explicit Computational Paths

Arthur F. Ramos, Ruy. J. G. B. de Queiroz, Anjolina G. de Oliveira and Tiago M. L. de Veras

Abstract

The treatment of equality as a type in type theory gives rise to an interesting type-theoretic structure known as 'identity type'. The idea is that, given terms a, b of a type A, one may form the type $Id_A(a, b)$, whose elements are proofs that a and b are equal elements of type A. A term of this type, $p : Id_A(a, b)$, makes up for the grounds (or proof) that establishes that a is indeed equal to b. Based on that, a proof of equality can be seen as a sequence of substitutions and rewrites, also known as a 'computational path'. One interesting fact is that it is possible to rewrite computational paths using a set of reduction rules arising from an analysis of redundancies in paths. These rules were mapped by De Oliveira in 1994 in a term rewrite system known as $LND_{EQ} - TRS$. Here we use computational paths and this term rewrite system to develop the main foundations of homotopy type theory, i.e., we develop the lemmas and theorems connected to the main types of this theory, types such as products, coproducts, identity type, transport and many others. We also show that it is possible to directly construct path spaces through computational paths. To show this, we use our path-based approach to construct two important structures of homotopy type theory: the path-space of natural numbers and the construction and calculation of the fundamental group of the circle.

Keywords. Computational paths, type theory, identity type, fundamental group of the circle, term rewriting systems.

1 Introduction

There seems to be little doubt that the identity type is one of the most intriguing concepts of Martin-Löf's Type Theory. This claim is supported by recent groundbreaking discoveries. In 2005, Vladimir Voevodsky [17] discovered the Univalent Models, resulting in a new area of research known as homotopy type theory [1]. This theory is based on the fact that a term of some identity type,

for example $p : Id_A(a, b)$, has a clear homotopical interpretation. The interpretation is that the witness p can be seen as a homotopical path between the points a and b within a topological space A. This simple interpretation has made clear the connection between type theory and homotopy theory, generating groundbreaking results, as one can see in [16, 1]. Nevertheless, it is important to emphasize that the homotopic paths exist only in the semantic sense. In other words, there is no formal entity in type theory that represents these paths. They are not present in the syntax of type theory.

In this work, we are interested in an entity known as computational path, originally proposed by [8]. A computational path is an entity that establishes the equality between two terms of the same type. It differs from the homotopical path, since it is not only a semantic interpretation. It is a formal entity of the equality theory. In fact, we proposed in [7, 14] that it should be considered as the type of the identity type. Moreover, we have further developed this idea in [15], where we proposed a groupoid model and proved that computational paths also refute the uniqueness of identity proofs. Thus, we obtained a result that is on par with the same one obtained by Hofmann & Streicher (1995) for the original identity type [12].

Our main idea in this work is to develop further our previous results. Specifically, we want to focus on the foundations of homotopy type theory. Our objective is to develop the main building blocks of this theory using computational paths. To do this, we prove quite a few lemmas and theorems of homotopy type theory involving the basic types, such as products, coproducts, transport, etc. We thus proceed to show that computational paths can be directly used to simulate path spaces. We argue that this is one of the main advantages of our approach, since it avoids the use of complicated techniques such as the code-encode-decode one. To illustrate that, we work with the natural numbers and with the fundamental group of the circle, showing how one can construct these structures through computational paths. These results also establish the foundations needed to calculate fundamental groups of many other structures. Indeed, we do these calculations in a recent and still unpublished work entitled *"On the Calculation of Fundamental Groups in homotopy type theory by Means of Computational Paths"*. A preprint version of that work can be found in [9].

This work is structured as thus: in sections 2, 3 and 4, we review the concept of computational paths and its connection to the identity type in type theory. In section 5, we use computational paths to establish the foundations of homotopy type theory. Since sections 2, 3 and 4 are only brief introductions to the theory of computational paths, we refer to papers [7] and [15] for a thorough introduction to this subject.

Computational Paths

Since computational path is a generic term, it is important to emphasize the fact that we are using the term computational path in the sense defined by [5]. A computational path is based on the idea that it is possible to formally define when two computational objects $a, b : A$ are equal. These two objects are equal if one can reach b from a applying a sequence of axioms or rules. This sequence of operations forms a path. Since it is between two computational objects, it is said that this path is a computational one. Also, an application of an axiom or a rule transforms (or rewrite) an term in another. For that reason, a computational path is also known as a sequence of rewrites. Nevertheless, before we define formally a computational path, we can take a look at one famous equality theory, the $\lambda\beta\eta - equality$ [11]:

Definition 1.1 *The* $\lambda\beta\eta$-equality *is composed by the following axioms:*

(α) $\lambda x.M = \lambda y.M[y/x]$ *if* $y \notin FV(M)$;

(β) $(\lambda x.M)N = M[N/x]$;

(ρ) $M = M$;

(η) $(\lambda x.Mx) = M$ $(x \notin FV(M))$.

And the following rules of inference:

(μ) $\dfrac{M = M'}{NM = NM'}$ (τ) $\dfrac{M = N \qquad N = P}{M = P}$

(ν) $\dfrac{M = M'}{MN = M'N}$ (σ) $\dfrac{M = N}{N = M}$

(ξ) $\dfrac{M = M'}{\lambda x.M = \lambda x.M'}$

Definition 1.2 *([11]) P is β-equal or β-convertible to Q (notation $P =_\beta Q$) iff Q is obtained from P by a finite (perhaps empty) series of β-contractions and reversed β-contractions and changes of bound variables. That is, $P =_\beta Q$ iff* ***there exist*** *$P_0, \ldots, P_n$ $(n \geq 0)$ such that $P_0 \equiv P$, $P_n \equiv Q$, $(\forall i \leq n-1)(P_i \triangleright_{1\beta} P_{i+1}$ or $P_{i+1} \triangleright_{1\beta} P_i$ or $P_i \equiv_\alpha P_{i+1})$.*

(NB: equality with an **existential** force, which will show in the proof rules for the identity type.)

The same happens with $\lambda\beta\eta$-equality:

Definition 1.3 *($\lambda\beta\eta$-equality [11]) The equality-relation determined by the theory $\lambda\beta\eta$ is called $=_{\beta\eta}$; that is, we define*

$$M =_{\beta\eta} N \quad \Leftrightarrow \quad \lambda\beta\eta \vdash M = N.$$

Example 1.4 *Take the term $M \equiv (\lambda x.(\lambda y.yx)(\lambda w.zw))v$. Then, it is $\beta\eta$-equal to $N \equiv zv$ because of the sequence:*
$(\lambda x.(\lambda y.yx)(\lambda w.zw))v, \quad (\lambda x.(\lambda y.yx)z)v, \quad (\lambda y.yv)z, \quad zv$
which starts from M and ends with N, and each member of the sequence is obtained via 1-step β- or η-contraction of a previous term in the sequence. To take this sequence into a path*, one has to apply transitivity twice, as we do in the example below.*

Example 1.5 *The term $M \equiv (\lambda x.(\lambda y.yx)(\lambda w.zw))v$ is $\beta\eta$-equal to $N \equiv zv$ because of the sequence:*
$(\lambda x.(\lambda y.yx)(\lambda w.zw))v, \quad (\lambda x.(\lambda y.yx)z)v, \quad (\lambda y.yv)z, \quad zv$
Now, taking this sequence into a path leads us to the following:
The first is equal to the second based on the grounds:
$\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v)$
The second is equal to the third based on the grounds:
$\beta((\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z)$
Now, the first is equal to the third based on the grounds:
$\tau(\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v), \beta((\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z))$
Now, the third is equal to the fourth one based on the grounds:
$\beta((\lambda y.yv)z, zv)$
Thus, the first one is equal to the fourth one based on the grounds:
$\tau(\tau(\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v), \beta((\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z)),$
$\beta((\lambda y.yv)z, zv)))$

The aforementioned theory establishes the equality between two λ-terms. Since we are working with computational objects as terms of a type, we need to translate the $\lambda\beta\eta$-equality to a suitable equality theory based on Martin Löf's type theory. We obtain:

Definition 1.6 *The equality theory of Martin Löf's type theory has the following basic proof rules for the Π-type:*

$$(\beta) \quad \frac{\begin{array}{c}[x:A]\\ N:A \qquad M:B\end{array}}{(\lambda x.M)N = M[N/x] : B[N/x]} \qquad (\xi) \quad \frac{\begin{array}{c}[x:A]\\ M = M' : B\end{array}}{\lambda x.M = \lambda x.M' : \Pi_{(x:A)}B}$$

$$(\rho) \quad \frac{M:A}{M = M : A} \qquad (\mu) \quad \frac{M = M' : A \qquad N : \Pi_{(x:A)}B}{NM = NM' : B[M/x]}$$

$$(\sigma)\quad \frac{M = N : A}{N = M : A} \qquad (\nu)\quad \frac{N : A \qquad M = M' : \Pi_{(x:A)}B}{MN = M'N : B[N/x]}$$

$$(\tau)\quad \frac{M = N : A \qquad N = P : A}{M = P : A}$$

$$(\eta)\quad \frac{M : \Pi_{(x:A)}B}{(\lambda x.Mx) = M : \Pi_{(x:A)}B}\ (x \notin FV(M))$$

We are finally able to formally define computational paths:

Definition 1.7 *Let a and b be elements of a type A. Then, a* computational path *s from a to b is a composition of rewrites (each rewrite is an application of the inference rules of the equality theory of type theory or is a change of bound variables). We denote that by $a =_s b$.*

As we have seen in example 1.5, composition of rewrites are applications of the rule τ. Since change of bound variables is possible, each term is considered up to α-equivalence.

2 Identity Type

In this section, we have two main objectives. The first one is to propose a formalization to the identity type using computational paths. The second objective is to show how can one use our approach to construct types representing reflexivity, transitivity and symmetry. In the case of the transitive type, we also compare our approach with the traditional one, i.e., Martin-Löf's Intensional type. With this comparison, we hope to show the clear advantage of our approach, in terms of simplicity. Since our approach is based on computational paths, we will sometimes refer to our formulation as the *path-based* approach and the traditional formulation as the *pathless* approach. By this we mean that, even though the homotopy type theory approach to the identity type brings about the notion of paths in the semantics, there is little in the way of handling paths as terms in the language of type theory.

Before the deductions that build the path-based identity type, we would like to make clear that we will use the following construction of the traditional approach [10]:

$$\frac{A \text{ type} \qquad a : A \qquad b : A}{Id_A(a,b) \text{ type}}\ Id - F \qquad\qquad \frac{a : A}{r(a) : Id_A(a,a)}\ Id - I$$

$$\frac{\begin{array}{cccc} a:A & b:A & c:Id_A(a,b) & \begin{array}{c}[x:A]\\ q(x):C(x,x,r(x))\end{array} \end{array} \qquad \begin{array}{c}[x:A, y:A, z:Id_A(x,y)]\\ C(x,y,z)\ \text{type}\end{array}}{J(p,q):C(a,b,c)}\ Id-E$$

2.1 Path-based construction

The best way to define any formal entity of type theory is by a set of natural deductions rules. Thus, we define our path-based approach as the following set of rules:

- Formation and Introduction rules:

$$\frac{A\ \text{type} \qquad a:A \qquad b:A}{Id_A(a,b)\ \text{type}}\ Id-F \qquad \frac{a =_s b:A}{s(a,b):Id_A(a,b)}\ Id-I$$

- Elimination rule:

$$\frac{\begin{array}{cc} m:Id_A(a,b) & \begin{array}{c}[a =_g b:A]\\ h(g):C\end{array}\end{array}}{REWR(m,\acute{g}.h(g)):C}\ Id-E$$

- Reduction rules:

$$\frac{\dfrac{a =_m b:A}{m(a,b):Id_A(a,b)}\ Id-I \qquad \begin{array}{c}[a =_g b:A]\\ h(g):C\end{array}}{\begin{array}{c}REWR(m,\acute{g}.h(g)):C\\ [a =_m b:A]\\ h(m/g):C\end{array}}\ Id-E \quad \rhd_\beta$$

$$\frac{e:Id_A(a,b) \qquad \dfrac{\dfrac{[a =_t b:A]}{t(a,b):Id_A(a,b)}\ Id-I}{Id-E}}{REWR(e,\acute{t}.t(a,b)):Id_A(a,b)} \quad \rhd_\eta \quad e:Id_A(a,b)$$

In these rules, $\acute{g}$ (and $\acute{t}$) to indicate that they are abstractions over the variable g (or t), for which the main rules of conversion of λ-abstraction hold. For that reason, we proposed two reduction rules that handle these conversions, the β and η reduction rules.

Our introduction and elimination rules reassures the concept of equality as an **existential force**. In the introduction rule, we encapsulate the idea that an witness of a identity type $Id_A(a, b)$ only exists if there exist a computational path establishing the equality of a and b. Also, the elimination rule is similar to the elimination rule of the existential quantifier. If we have an witness for $Id_A(a, b)$, and if from a computational path between a and b we can construct a term of type C, then we can eliminate the identity type, obtaining a term of type C.

3 A Term Rewriting System for Paths

As we have just shown, a computational path establishes when two terms of the same type are equal. From the theory of computational paths, an interesting case arises. Suppose we have a path s that establishes that $a =_s b : A$ and a path t that establishes that $a =_t b : A$. Consider that s and t are formed by distinct compositions of rewrites. Is it possible to conclude that there are cases that s and t should be considered equivalent? The answer is *yes*. Consider the following example:

Example 3.1 Consider the path $a =_t b : A$. By the symmetric property, we obtain $b =_{\sigma(t)} a : A$. What if we apply the property again on the path $\sigma(t)$? We would obtain a path $a =_{\sigma(\sigma(t))} b : A$. Since we applied symmetry twice in succession, we obtained a path that is equivalent to the initial path t. For that reason, we conclude the act of applying symmetry twice in succession is a redundancy. We say that the path $\sigma(\sigma(t))$ can be reduced to the path t.

As one could see in the aforementioned example, different paths should be considered equal if one is just a redundant form of the other. The example that we have just seen is just a straightforward and simple case. Since the equality theory has a total of 7 axioms, the possibility of combinations that could generate redundancies is high. Fortunately, most redundancies were thoroughly mapped by [2]. In this work, a system that establishes redundancies and creates rules that solve them was proposed. This system, known as $LND_{EQ} - TRS$, mapped a total of 39 rules that solve redundancies. These 39 rules can be checked in **Appendix B**. For each rule, there is a proof tree that constructs it. All proof trees can be checked in [7]. In the case of example **3**, we have the following [7]:

$$\dfrac{\dfrac{x =_t y : A}{y =_{\sigma(t)} x : A}}{x =_{\sigma(\sigma(t))} y : A} \quad \rhd_{ss} \quad x =_t y : A$$

It is important to notice that we assign a label to every rule. In the previous case, we assigned the label *ss*.

Definition 3.2 An *rw*-rule is any of the rules defined in $LND_{EQ} - TRS$.

Definition 3.3 *Let s and t be computational paths. We say that $s \rhd_{1rw} t$ (read as: s rw-contracts to t) iff we can obtain t from s by an application of only one rw-rule. If s can be reduced to t by finite number of rw-contractions, then we say that $s \rhd_{rw} t$ (read as s rw-reduces to t).*

Definition 3.4 Let s and t be computational paths. We say that $s =_{rw} t$ (read as: s is *rw*-equal to t) iff t can be obtained from s by a finite (perhaps empty) series of *rw*-contractions and reversed *rw*-contractions. In other words, $s =_{rw} t$ iff there exists a sequence $R_0, ..., R_n$, with $n \geq 0$, such that

$$(\forall i \leq n-1)(R_i \rhd_{1rw} R_{i+1} \text{ or } R_{i+1} \rhd_{1rw} R_i)$$
$$R_0 \equiv s, \quad R_n \equiv t$$

Proposition 3.5 *is transitive, symmetric and reflexive.*

Proof. Comes directly from the fact that *rw*-equality is the transitive, reflexive and symmetric closure of *rw*. ■

We'd like to mention that $LND_{EQ} - TRS$ is terminating and confluent. The proof of this affirmation can be found in [2, 4, 3, 6].

Thus, we conclude our review of computational paths as terms of the identity type and the associated rewrite system. If necessary, please check [7] and [15] for a thorough development of this theory.

4 Homotopy Type Theory

In the previous sections, we have said that one of the most interesting concepts of type theory is the identity type. We have also said that the reason for that is the fact one can see the identity type as a homotopical path between two points of a space, giving rise to a homotopical interpretation of type theory. The connection between those two theories created a whole new area of research known as homotopy type theory. In this work, we introduced computational paths as the syntactic counterpart of those homotopical paths, since they only

exist in a semantical sense. Nevertheless, we have not talked yet how one can use computational paths in homotopy type theory. Thus, in this section, we develop the main objective of this work.

We want to show that some of the foundational definitions, propositions and theorems of homotopy type theory still hold in our path-based approach. In other words, we use our approach to construct the building blocks of more complex results.

One important fact to notice is that every proof that does not involve the identity type is valid in the path-based approach. This is obvious, since the only difference between the traditional approach and ours is the formulation of the identity type. If a proof uses it, we need to reformulate this proof using our path-based approach, instead of using the induction principle of the traditional one. Thus, every part of a proof that is not directly or indirectly related to identity type is still valid in our approach.

In a path-based proof, we are going to use the formulation proposed in the previous sections. We also are going to use the reduction rules of $LND_{EQ} - TRS$. In the process of developing the theory of this section, we noticed that $LND_{EQ} - TRS$, as proposed in the previous section is still incomplete. We state this based on the fact that we found new reduction rules that are not part of the original $LND_{EQ} - TRS$. That way, we added these new rules to the system, expanding it.

4.1 Groupoid Laws

In our previous work [15], we have seen that computational paths form a groupoid structure. Let's check again those rules using our $REWR$ constructor directly:

Lemma 4.1 *The type* $\Pi_{(a:A)} Id_A(a,a)$ *is inhabited.*

Proof. We construct an witness for the desired type:

$$\dfrac{\dfrac{\dfrac{[a:A]}{a =_\rho a : A}}{\rho(a,a) : Id_A(a,a)}\, Id - I_1}{\lambda a.\rho(a,a) : \Pi_{(a:A)} Id_A(a,a)}\, \Pi - I$$

■

Lemma 4.2 *The type* $\Pi_{(a:A)}\Pi_{(b:A)}(Id_A(a,b) \to Id_A(b,a))$ *is inhabited.*

Proof. Similar to the previous lemma, we construct an witness:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[a:A]\quad[b:A]\qquad\dfrac{\dfrac{[a =_t b : A]}{b =_{\sigma(t)} a : A}}{(\sigma(t))(b,a) : Id_A(b,a)}\,Id-I}{[p(a,b) : Id_A(a,b)]}\,Id-E}{REWR(p(a,b), \acute{t}.(\sigma(t))(b,a)) : Id_A(b,a)}\,\Pi-I}{\lambda p.REWR(p(a,b), \acute{t}.(\sigma(t))(b,a)) : Id_A(a,b) \to Id_A(b,a)}\,\Pi-I}{\lambda b.\lambda p.REWR(p(a,b), \acute{t}.(\sigma(t))(b,a)) : \Pi_{(b:A)}(Id_A(a,b) \to Id_A(b,a))}\,\Pi-I}{\lambda a.\lambda b.\lambda p.REWR(p(a,b), \acute{t}.(\sigma(t))(b,a)) : \Pi_{(a:A)}\Pi_{(b:A)}(Id_A(a,b) \to Id_A(b,a))}\,\Pi-I$$

■

Lemma 4.3 *The type* $\Pi_{(a:A)}\Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a,b) \to Id_A(b,c) \to Id_A(a,c))$ *is inhabited.*

Proof. We construct the witness in Figure 4.1.

■

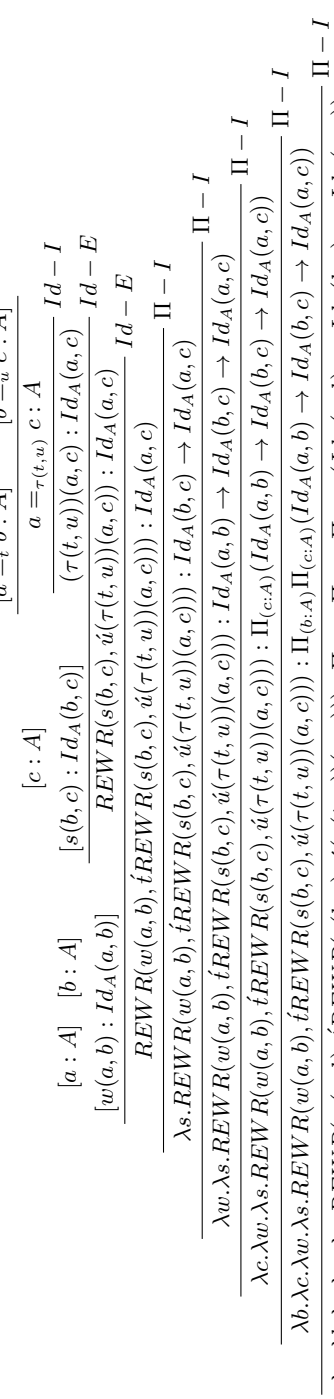

Figure 1: Lemma's 4.3 inhabitant

Lemmas **1**, **2** and **3** correspond respectively to the reflexivity, symmetry and transitivity of the identity type. From now on, the reflexivity will be represented by ρ, symmetry by σ and transitivity by τ.

Lemma 4.4 *For any type A, $x, y, z, w : A$ and $p : Id_A(x, y)$ and $q : Id_A(y, z)$ and $r : Id_A(z, w)$, the following types are inhabited:*

1. $\Pi_{(x,y:A)}\Pi_{(p:Id_A(x,y))}Id_{Id_A(x,y)}(p, \rho_y \circ p)$ *and* $\Pi_{(x,y:A)}\Pi_{(p:Id_A(x,y))}Id_{Id_A(x,y)}(p, p \circ \rho_x)$.
2. $\Pi_{(x,y:A)}\Pi_{(p:Id_A(x,y))}Id_{Id_A(x,y)}(\sigma(p) \circ p, \rho_x)$ *and* $\Pi_{(x,y:A)}\Pi_{(p:Id_A(x,y))}Id_{Id_A(x,y)}(p \circ \sigma(p), \rho_y)$
3. $\Pi_{(x,y:A)}\Pi_{(p:Id_A(x,y))}Id_{Id_A(x,y)}(\sigma(\sigma(p)), p)$
4. $\Pi_{(x,y,z,w:A)}\Pi_{(p:Id_A(x,y))}\Pi_{(q:Id_A(y,z))}\Pi_{(r:Id_A(z,w))}Id_{Id_A(x,w)}(r \circ (q \circ p), (r \circ q) \circ p)$

Proof. The proof of each statement follows from the same idea. We just need to look for suitable reduction rules already present in the original $LND_{EQ} - TRS$.

1. The first thing to notice is that a composition in our path-based approach corresponds to a transitive operation, i.e., $(p \circ \rho_x)$ can be written as $\tau(\rho_x, p)$ Follows from rules number **5** and **6**. These are as follows:

$$\frac{x =_r y : A \qquad y =_\rho y : A}{x =_{\tau(r,\rho)} y : A} \quad \rhd_{trr} \quad x =_r y : A$$

$$\frac{x =_\rho x : A \qquad x =_r y : A}{x =_{\tau(\rho,r)} y : A} \quad \rhd_{tlr} \quad x =_r y : A$$

Thus, we have:

$$\frac{\dfrac{\tau(p, \rho_y) =_{trr} p : Id_A(x, y)}{(trr)(\tau(p, \rho_y), p) : Id_{Id_A(x,y)}(p, \rho_y \circ p)}}{\lambda x.\lambda y.\lambda p.(trr)(\tau(p, \rho_y), p) : \Pi_{(x,y:A)}\Pi_{(p:Id_A(x,y))}Id_{Id_A(x,y)}(p, \rho_y \circ p)}$$

$$\frac{\dfrac{\tau(\rho_x,p) =_{tlr} p : Id_A(x,y)}{(tlr)(\tau(\rho_x,p),p) : Id_{Id_A(x,y)}(p,p\circ\rho_x)}}{\lambda x.\lambda y.\lambda p.(tlr)(\tau(\rho_x,p),p) : \Pi_{(x,y:A)}\Pi_{(p:Id_A(x,y))}Id_{Id_A(x,y)}(p,p\circ\rho_x)}$$

2. We use rules **3** and **4**:

$$\frac{x =_r y : A \qquad y =_{\sigma(r)} x : A}{x =_{\tau(r,\sigma(r))} x : A} \quad \rhd_{tr} \quad x =_\rho x : A$$

$$\frac{y =_{\sigma(r)} x : A \qquad x =_r y : A}{y =_{\tau(\sigma(r),r)} y : A} \quad \rhd_{tsr} \quad y =_\rho y : A$$

Thus:

$$\frac{\dfrac{\tau(p,\sigma(p)) =_{tr} \rho_x : Id_A(x,y)}{(tr)(\tau(p,\sigma(p)),\rho_x) : Id_{Id_A(x,y)}(\sigma(p)\circ p,\rho_x)}}{\lambda x.\lambda y.\lambda p.(tr)(\tau(p,\sigma(p),\rho_x) : \Pi_{(x,y:A)}\Pi_{(p:Id_A(x,y))}Id_{Id_A(x,y)}(\sigma(p)\circ p,\rho_x)}$$

$$\frac{\dfrac{\tau(\sigma(p),p) =_{tsr} \rho_y : Id_A(x,y)}{(tsr)(\tau(\sigma(p),p),\rho_y) : Id_{Id_A(x,y)}(p\circ\sigma(p),\rho_y)}}{\lambda x.\lambda y.\lambda p.(tsr)(\tau(p,\sigma(p),\rho_y) : \Pi_{(x,y:A)}\Pi_{(p:Id_A(x,y))}Id_{Id_A(x,y)}(p\circ\sigma(p),\rho_y)}$$

3. We use rule **2**:

$$\frac{\dfrac{x =_r y : A}{y =_{\sigma(r)} x : A}}{x =_{\sigma(\sigma(r))} y : A} \quad \rhd_{ss} \quad x =_r y : A$$

Thus:

$$\dfrac{\dfrac{\sigma(\sigma(p)) =_{ss} p : Id_A(x,y)}{(ss)(\sigma(\sigma(p),p)) : Id_{Id_A(x,y)}(\sigma(\sigma(p)),p)}}{\lambda x.\lambda y.\lambda p.(ss)(\sigma(\sigma(p),p)) : \Pi_{(x,y:A)}\Pi_{(p:Id_A(x,y))}Id_{Id_A(x,y)}(\sigma(\sigma(p)),p)}$$

4. We use rule **37**:

$$\dfrac{\dfrac{x =_t y : A \qquad y =_r w : A}{x =_{\tau(t,r)} w : A} \qquad w =_s z : A}{x =_{\tau(\tau(t,r),s)} z : A}$$

$$\rhd_{tt} \dfrac{x =_t y : A \qquad \dfrac{y =_r w : A \qquad w =_s z : A}{y =_{\tau(r,s)} z : A}}{x =_{\tau(t,\tau(r,s))} z : A}$$

Thus:

$$\dfrac{\dfrac{\tau(\tau(p,q),r) =_{tt} \tau(p,\tau(q,r)) : Id_A(x,w)}{(tt)(\tau(\tau(p,q),r) =_{tt} \tau(p,\tau(q,r))) : Id_{Id_A(x,w)}(r\circ(q\circ p),(r\circ q)\circ p)}}{M : \Pi_{(p:Id_A(x,y))}\Pi_{(q:Id_A(y,z))}\Pi_{(r:Id_A(z,w))}Id_{Id_A(x,w)}(r\circ(q\circ p),(r\circ q)\circ p)}$$

where $M = \lambda x.\lambda y.\lambda z.\lambda w.\lambda p.\lambda q.\lambda r.(ss)(\sigma(\sigma(p),p))$

■

With the previous lemma, we showed that our path-based approach yields the groupoid structure of a type up to propositional equality.

4.2 Functoriality

We want to show that functions preserve equality[16].

Lemma 4.5 *The type* $\Pi_{(x,y:A)}\Pi_{(f:A\to B)}(Id_A(x,y) \to Id_B(f(x),f(y)))$ *is inhabited.*

Proof. It is a straightforward construction:

$$\dfrac{\dfrac{\dfrac{[x =_s y : A] \qquad [f : A \to B]}{f(x) =_{\mu_f(s)} f(y) : B}}{\mu_f(s)(f(x), f(y)) : Id_B(f(x), f(y))} \qquad [p : Id_A(x, y)]}{\dfrac{REWR(p, \lambda s.\mu_f(s)(f(x), f(y))) : Id_B(f(x), f(y))}{M : \Pi_{(x,y:A)}\Pi_{(f:A\to B)}(Id_A(x, y) \to Id_B(f(x), f(y)))}}$$

where $M = \lambda x.\lambda y.\lambda f.\lambda p.REWR(p, \lambda s.\mu_f(s)(f(x), f(y)))$ ■

Lemma 4.6 *For any functions $f : A \to B$ and $g : B \to C$ and paths $p : x =_A y$ and $q : y =_A z$, we have:*

1. $\mu_f(\tau(p, q)) = \tau(\mu_f(p), \mu_f(q))$
2. $\mu_f(\sigma(p)) = \sigma(\mu_f(p))$
3. $\mu_g(\mu_f(p)) = \mu_{g\circ f}(p)$
4. $\mu_{Id_A}(p) = p$

Proof.

1. For the first time, we need to add a new rule to the original 39 rules of $LND_{EQ} - TRS$. We introduce rule **40**:

$$\dfrac{\dfrac{x =_p y : A \qquad [f : A \to B]}{f(x) =_{\mu_f(p)} f(y) : B} \qquad \dfrac{y =_q z : A \qquad [f : A \to B]}{f(y) =_{\mu_f(q)} f(z) : B}}{f(x) = \tau(\mu_f(p), \mu_f(q))f(z) : B}$$

$$\rhd_{tf} \dfrac{\dfrac{x =_p y : A \qquad y =_q z : A}{x =_{\tau(p,q)} z : A} \qquad f : A \to B}{f(x) =_{\mu_f(\tau(p,q))} f(z) : B}$$

   Thus, we have $\mu_f(\tau(p, q)) =_{\sigma(tf)} \tau(\mu_f(p), \mu_f(q))$

2. This one follows from rule **30**:

$$\dfrac{\dfrac{x =_p y : A \qquad [f : A \to B]}{f(x) =_{\mu_f(p)} f(y) : B}}{f(y) =_{\sigma(\mu_f(p))} f(x) : B}$$

$$\triangleright_{sm} \dfrac{\dfrac{x =_p y : A}{y =_{\sigma(p)} x : A} \qquad [f : A \to B]}{f(y) =_{\mu_f(\sigma(p))} f(x) : B}$$

We have $\mu_f(\sigma(p)) =_{\sigma(sm)} \sigma(\mu_f(p))$

3. We introduce rule **41**:

$$\dfrac{\dfrac{x =_p y : A \qquad [f : A \to B]}{f(x) =_{\mu_f(p)} f(y) : B} \qquad [g : B \to C]}{g(f(x)) =_{\mu_g(\mu_f(p))} g(f(y)) : C}$$

$$\triangleright_{cf} \dfrac{x =_p y : A \qquad \dfrac{\dfrac{[x : A] \qquad [f : A \to B]}{f(x) : B} \qquad [g : B \to C]}{g(f(x)) : C}}{\lambda x.g(f(x)) \equiv (g \circ f) : A \to C}$$
$$g(f(x)) =_{\mu_{g \circ f}(p)} g(f(y)) : C$$

Then, $\mu_g(\mu_f(p)) =_{cf} \mu_{g \circ f}(p)$

4. We introduce rule **42**:

$$\dfrac{\dfrac{x =_p y : A \qquad [Id_A : A \to A]}{Id_A(x) = \mu_{Id_A(p)} Id_A(y) : A}}{x =_{\mu_{Id_A}(p)} y : A} \quad \triangleright_{ci} \quad x =_p y : A$$

It follows that $\mu_{Id_A}(p) =_{ci} p$

■

4.3 Transport

As stated in [5], substitution can take place when no quantifier is involved. In this sense, there is a 'quantifier-less' notion of substitution. In type theory, this 'quantifier-less' substitution is given by a operation known as transport [16]. In our path-based approach, we formulate a new inference rule of 'quantifier-less' substitution [5]:

$$\frac{x =_p y : A \qquad f(x) : P(x)}{p(x,y) \circ f(x) : P(y)}$$

We use this transport operation to solve one essential issue of our path-based approach. We know that given a path $x =_p y : A$ and function $f : A \to B$, the application of axiom μ yields the path $f(x) =_{\mu_f(p)} f(y) : B$. The problem arises when we try to apply the same axiom for a dependent function $f : \Pi_{(x:A)} P(x)$. In that case, we want $f(x) = f(y)$, but we cannot guarantee that the type of $f(x) : P(x)$ is the same as $f(y) : P(y)$. The solution is to apply the transport operation and thus, we can guarantee that the types are the same:

$$\frac{x =_p y : A \qquad f : \Pi_{(x:A)} P(x)}{p(x,y) \circ f(x) =_{\mu_f(p)} f(y) : P(y)}$$

Lemma 4.7 *(Leibniz's Law) The type* $\Pi_{(x,y:A)}(Id_A(x,y) \to P(x) \to P(y))$ *is inhabited.*

Proof. We construct the following tree:

$$\dfrac{\dfrac{\dfrac{[x =_p y : A] \qquad [f(x) : P(x)]}{p(x,y) \circ f(x) : P(y)}}{\lambda f(x).p(x,y) \circ f(x) : P(x) \to P(y)} \qquad [z : Id_A(x,y)]}{\dfrac{REWR(z, \lambda p.\lambda f(x).p(x,y) \circ f(x)) : P(x) \to P(y)}{M : \Pi_{(x,y:A)}(Id_A(x,y) \to P(x) \to P(y))}}$$

where $M = \lambda x.\lambda y.\lambda z.REWR(z, \lambda p.\lambda f(x).p(x,y) \circ f(x))$ ■

The function $\lambda f(x).p(x,y) \circ f(x) : P(x) \to P(y)$ is usually written as $transport^p(p, -)$ and $transport^p(p, f(x)) : P(y)$ is usually written as $p_*(f(x))$.

Lemma 4.8 *For any $P(x) \equiv B$, $x =_p y : A$ and $b : B$, there is a path $transport^P(p, b) = b$.*

Proof. The first to notice is the fact that in our formulation of transport, we always need a functional expression $f(x)$, and in this case we have only a constant term b. To address this problem, we consider a function $f = \lambda.b$ and then, we transport over $f(x) \equiv b$:

$$transport^P(p, f(x) \equiv b) =_{\mu(p)} (f(y) \equiv b).$$

Thus, $transport^P(p, b) =_{\mu(p)} b$. We may call this path $transportconst^B_p(b)$. ■

Lemma 4.9 *For any $f : A \to B$ and $x =_p y : A$, we have*

$$\mu(p)(p_*(f(x)), f(y)) = \tau(transportconst^B_p, \mu_f(p))(p_*(f(x)), f(y))$$

Proof. The first thing to notice is that in this case, $transportconst^B_p$ is the path $\mu(p)(p * (f(x), f(x))$ by Lemma **8**. As we did to the rules of $LND_{EQ} - TRS$, we establishes this equality by getting to the same conclusion from the same premises by two different trees:

In the first tree, we consider $f(x) \equiv b : B$ and transport over $b : B$:

$$\dfrac{\dfrac{\dfrac{x =_p y : A \qquad f(x) \equiv b : B}{p(x,y) \circ (f(x) \equiv b) : B}}{p_*(f(x)) =_{\mu_f(p)} b \equiv f(x)} \qquad \dfrac{x =_p y : A \qquad f : A \to B}{f(x) =_{\mu_f(p)} f(y) : B}}{p_*(f(x)) =_{\tau(\mu_f(p), \mu_f(p))} f(y) : B}$$

In the second one, we consider $f(x)$ as an usual functional expression and thus, we transport the usual way:

$$\dfrac{\dfrac{x =_p y : A \qquad f(x) : B}{p(x,y) \circ f(x) : B}}{p_*(f(x)) =_{\mu_f(p)} f(y) : B}$$

■

Lemma 4.10 *For any $x =_p y : A$ and $q : y =_A z : A$, $f(x) : P(x)$, we have*

$$q_*(p_*(f(x))) = (p \circ q)_*(f(x))$$

Proof. We develop both sides of the equation and wind up with the same result:

$$\begin{gathered} q_*(p_* f(x)) =_{\mu(p)} q_*(f(y)) =_{\mu(q)} f(z) \\ (p \circ q)_*(f(x)) =_{\mu(p \circ q)} f(z) \end{gathered}$$

■

Lemma 4.11 *For any $f : A \to B$, $x =_p y : A$ and $u : P(f(x))$, we have:*

$$transport^{P \circ f}(p, u) = transport^{P}(\mu_f(p), u)$$

Proof. This lemma hinges on the fact that there is two possible interpretations of u that stems from the fact that $(g \circ f)(x) \equiv g(f(x))$. Thus, we can see u as functional expression g on $f(x)$ or an expression $g \circ f$ on x (cf. Figure 2).
■

$$\dfrac{\dfrac{x =_p y : A \qquad u \equiv (g \circ f)(x) : (P \circ f)(x)}{p(x,y) \circ (g \circ f)(x) : (P \circ f)(y)} \qquad \dfrac{\dfrac{x =_p y : A}{f(x) =_{u_f(p)} f(y) : B} \qquad u \equiv g(f(x)) : P(f(x))}{\mu_f(p)(f(x), f(y)) \circ g(f(x)) : P(f(y))}}{\dfrac{p(x,y) \circ (g \circ f)(x) =_{\mu(p)} (g \circ f)(y) : (P \circ f)(y)}{p(x,y) \circ (g \circ f)(x) =_{\mu(p)} g(f(y)) : P(f(y))} \qquad \dfrac{\mu_f(p)(f(x), f(y)) \circ g(f(x)) =_{\mu(p)} g(f(y)) : P(f(y))}{g(f(y)) =_{\sigma(\mu(p))} \mu_f(p)(f(x), f(y)) \circ g(f(x)) : P(f(y))}}{\dfrac{p(x,y) \circ (g \circ f)(x) =_{\tau(\mu(p),\sigma(\mu(p)))} \mu_f(p)(f(x), f(y)) \circ g(f(x)) : P(f(y))}{transport^{P \circ f}(p, u) =_{\tau(\mu(p),\sigma(\mu(p)))} transport^P(\mu_f(p), u)}}$$

Figure 2: Statement Lemma 4.11

Lemma 4.12 *For any* $f : \Pi_{(x:A)}P(x) \to Q(x)$, $x =_p y : A$ *and* $u(x) : P(x)$, *we have:*

$$transport^Q(p, f(u(x))) = f(transport^P(p, u(x)))$$

Proof. We proceed the usual way, constructing a derivation tree that establishes the equality:

$$\dfrac{\dfrac{\dfrac{x =_p y : A \quad f(u(x)) : Q(x)}{p(x,y) \circ f(u(x)) : Q(y)}}{p(x,y) \circ f(u(x)) =_{\mu(p)} f(u(y)) : Q(y)} \quad \dfrac{\mathcal{D}}{f(u(y)) =_{\sigma(\mu_f(\mu(p)))} f(p(x,y) \circ u(x)) : Q(y)}}{\dfrac{p(x,y) \circ f(u(x)) =_{\tau(\mu(p),\sigma(\mu_f(\mu(p))))} f(p(x,y) \circ u(x))}{transport^Q(p, f(u(x))) =_{\tau(\mu(p),\sigma(\mu_f(\mu(p))))} f(transport^P(p, u(x)))}}$$

where $\mathcal{D}$ is

$$\dfrac{\dfrac{\dfrac{x =_p y : A \quad u(x) : P(x)}{p(x,y) \circ u(x) : P(y)}}{p(x,y) \circ u(x) =_{\mu(p)} u(y) : P(y)} \quad f : \Pi_{(x:A)}P(x) \to Q(x)}{\dfrac{f(p(x,y) \circ u(x)) =_{\mu_f(\mu(p))} f(u(y)) : Q(y)}{f(u(y)) =_{\sigma(\mu_f(\mu(p)))} f(p(x,y) \circ u(x)) : Q(y)}}$$

■

4.4 Homotopies

In homotopy type theory, a homotopy is defined as follows [16]:

Definition 4.13 *For any* $f, g : \Pi_{(x:A)}P(x)$, *a homotopy from* f *to* g *is a dependent function of type:*

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

In our path-based approach, we have a homotopy $f, g : \Pi_{(x:A)}P(x)$ if for every $x : A$ we have a computational path between $f(x) = g(x)$. Thus, if we have a homotopy $H_{f,g} : f \sim g$, we derive the following rule:

$$\frac{H_{f,g} : f \sim g \qquad f, g : \Pi_{(x:A)} P(x) \qquad x : A}{f(x) =_{H_{f,g}(x)} g(x) : P(x)}$$

And:

$$\frac{f, g : \Pi_{(x:A)} P(x) \qquad x : A \qquad \overset{[f,g:\Pi_{(x:A)}P(x), x:A]}{f(x) =_p g(x)}}{H^p_{f,g} : f \sim g}$$

Lemma 4.14 *For any $f, g, h : A \to B$, the following types are inhabited:*

1. $f \sim f$
2. $(f \sim g) \to (g \sim f)$
3. $(f \sim g) \to (g \sim h) \to (f \sim h)$

Proof.

1. We construct the following term:

$$\frac{f : A \to B \qquad x : A \qquad \dfrac{\dfrac{[x : A]}{x =_\rho x} \qquad [f : A \to B]}{f(x) =_{\mu_f(\rho)} f(x) : B}}{H^{\mu_f(\rho)}_{f,f} : f \sim f}$$

2. We construct:

$$\frac{\dfrac{f, g : A \to B \quad x : A \qquad \dfrac{\dfrac{[H_{f,g} : f \sim g] \quad [f, g : A \to B] \quad [x : A]}{f(x) =_{H_{f,g}(x)} g(x) : B}}{g(x) =_{\sigma(H_{f,g}(x))} f(x) : B}}{H^{\sigma(H_{f,g}(x))}_{g,f} : g \sim f}}{\lambda H_{f,g}.H^{\sigma(H_{f,g}(x))}_{g,f} : (f \sim g) \to (g \sim f)}$$

3. We construct:

$$\dfrac{\dfrac{f,h : A \to B \qquad x : A \qquad \dfrac{\mathcal{D}}{f(x) =_{\tau(H_{f,g}(x),H_{g,z}(x))} h(x) : B}}{H_{f,h}^{\tau(H_{f,g}(x),H_{g,z}(x))} : f \sim h}}{\lambda H_{f,g}.\lambda H_{g,h}.H_{f,h}^{\tau(H_{f,g}(x),H_{g,z}(x))} : (f \sim g) \to (g \sim h) \to (f \sim h)}$$

where $\mathcal{D}$ is

$$\dfrac{\dfrac{[H_{f,g} : f \sim g] \quad [f,g : A \to B] \quad [x : A]}{f(x) =_{H_{f,g}(x)} g(x) : B} \qquad \dfrac{[H_{g,h} : g \sim h] \quad [g,h : A \to B] \quad [x : A]}{g(x) =_{H_{g,h}(x)} h(x) : B}}{f(x) =_{\tau(H_{f,g}(x),H_{g,z}(x))} h(x) : B}$$

■

Lemma 4.15 *For any $H_{f,g} : f \sim g$ and functions $f, g : A \to B$ and a path $x =_p y : A$ we have:*

$$\tau(H_{f,g}(x), \mu_g(p)) = \tau(\mu_f(p), H_{f,g}(y))$$

Proof.

To establish this equality, we need to add a new rule to our $LND_{EQ}-TRS$. We introduce rule **43**:

$$\dfrac{\dfrac{H_{f,g} : f \sim g \qquad x : A \qquad f,g : A \to B}{f(x) =_{H_{f,g}(x)} g(x) : B} \qquad \dfrac{x =_p y : A}{g(x) =_{\mu_g(p)} g(y) : B}}{f(x) =_{\tau(H_{f,g}(x),\mu_g(p))} g(y) : B}$$

$$\rhd_{hp}$$

$$\dfrac{\dfrac{x =_p y : A}{f(x) =_{\mu_f(p)} f(y) : B} \qquad \dfrac{H_{f,g} : f \sim g \qquad x : A \qquad f,g : A \to B}{f(y) =_{H_{f,g}(y)} g(y) : B}}{f(x) =_{\tau(\mu_f(p),H_{f,g}(y)} g(y) : B}$$

And thus:

$$\tau(H_{f,g}(x), \mu_g(p)) =_{hp} \tau(\mu_f(p), H_{f,g}(y))$$

■

After this section, we start to study specific lemmas and theorems involving basic types of type theory. Nevertheless, several of those theorems are statements about the notion of **equivalence** (notation: $\simeq$). Before we define equivalence, we need the following definition [16]:

Definition 4.16 *A **quasi-inverse** of a function $f : A \to B$ is a triple (g, α, β) such that g is a function $g : B \to A$ and α and β are homotopies such that $\alpha : f \circ g \sim Id_B$ and $\beta : g \circ f \sim Id_A$*

A quasi-inverse of f is usually written as $qinv(f)$.

Definition 4.17 *A function $f : A \to B$ is an equivalence if there is a quasi-inverse $qinv(f) : B \to A$.*

4.5 Cartesian Product

We start proving some important lemmas and theorems for the Cartesian product type. As we did in previous subsections, we proceed using our path-based approach. Before we prove our first theorem, it is important to remember that given a term $x : A \times B$, we can extract two projections, $FST(x) : A$ and $SND(x) : B$. Thus, given a path $x =_p y : A \times B$, we extract paths $FST(x) = SND(y) : A$ and $SND(x) = SND(y) : B$.

Theorem 4.18 *The function $(x =_p y : A \times B) \to (FST(x) = FST(y) : A) \times (SND(x) = SND(y) : B)$ is an equivalence for any x and y.*

Proof. To show the equivalence, we need to show the following

1. From $x =_p y : A \times B$ we want to obtain $(FST(x) = FST(y) : A) \times (SND(x) = SND(y) : B)$ and from that, we want to go back to $x =_p y : A \times B$.
2. We want to do the inverse process. From $(FST(x) = FST(y) : A) \times (SND(x) = SND(y) : B)$ we want to obtain $x =_p y : A \times B$ and then go back to $(FST(x) = FST(y) : A) \times (SND(x) = SND(y) : B)$.

To show the first part, we need rule **21**:

$$\frac{\dfrac{x =_p y : A \times B}{FST(x) =_{\mu_1(p)} FST(y) : A} \qquad \dfrac{x =_p y : A \times B}{SND(x) =_{\mu_2(p)} SND(y) : B}}{\langle FST(x), SND(x)\rangle =_{\epsilon(\mu_1(p),\mu_2(p))} \langle FST(y), SND(y)\rangle : A \times B}$$

$$\rhd_{mx}\ x =_p y : A \times B.$$

Thus, applying rule mx we showed the first part of our proof. For the second part, we need rules **14** and **15**:

$$\frac{\dfrac{x =_r x' : A \qquad y =_s z : B}{\langle x, y\rangle =_{\epsilon_\wedge(r,s)} \langle x', z\rangle : A \times B}}{FST(\langle x, y\rangle) =_{\mu_1(\epsilon_\wedge(r,s))} FST(\langle x', z\rangle) : A}$$

$$\rhd_{mx2l}\ x =_r x' : A.$$

And:

$$\frac{\dfrac{x =_r y : A \qquad z =_s w : B}{\langle x, z\rangle =_{\epsilon_\wedge(r,s)} \langle y, w\rangle : A \times B}}{FST(\langle x, z\rangle) =_{\mu_2(\epsilon_\wedge(r,s))} FST(\langle y, w\rangle) : B}$$

$$\rhd_{mx2r}\ z =_s w : B.$$

We also use the η-reduction for the Cartesian product:

$$\langle FST(x), SND(x)\rangle : A \times B \rhd_\eta x : A \times B$$

We construct the following derivation tree:

$$\dfrac{\dfrac{\dfrac{\langle FST(x) =_s FST(y), SND(x) =_t SND(y)\rangle}{FST(x) =_s FST(y) : A} \qquad \dfrac{\langle FST(x) =_s FST(y), SND(x) =_t SND(y)\rangle}{SND(x) =_t SND(y) : B}}{\langle FST(x), SND(x)\rangle =_{\epsilon_\wedge(s,t)} \langle FST(y), SND(y)\rangle : A \times B}}{x =_{\epsilon(s,t)} y : A \times B}\ \rhd_\eta$$

From $x =_{\epsilon(s,t)} y : A \times B$, we have:

$$\dfrac{\dfrac{\dfrac{x =_{\epsilon(s,t)} y : A \times B}{FST(x) =_{\mu_1(\epsilon_\wedge(s,t)} FST(y) : A} \quad \dfrac{x =_{\epsilon(s,t)} y : A \times B}{SND(x) =_{\mu_2(\epsilon_\wedge(s,t)} SND(y) : B}}{\langle FST(x) =_{\mu_1(\epsilon_\wedge(s,t)} FST(y), SND(x) =_{\mu_2(\epsilon_\wedge(s,t)} SND(y)\rangle}\wedge - I}{\langle FST(x) =_s FST(y), SND(x) =_t SND(y)\rangle}\rhd_{mx2l,mx2r}$$

Thus, we showed part 2 and concluded the proof of this theorem. ■

Theorem 4.19 *For any $x, y : A \times B$, $FST(x) =_p FST(y) : A$, $SND(x) =_q SND(y) : B$, functions $g : A \to A'$, $h : B \to B'$ and $f : A \times B \to A' \times B'$ defined by $f(x) \equiv \langle g(FST(x)), h(SND(x)\rangle$, we have:*

$$\mu_f(\epsilon_\wedge(p, q)) = \epsilon_\wedge(\mu_g(p), \mu_h(q))$$

Proof. We introduce rule **44**:

$$\dfrac{\dfrac{\dfrac{FST(x) =_p FST(y) : A \qquad SND(x) =_q SND(y) : B}{\langle FST(x), SND(x)\rangle =_{\epsilon_\wedge(p,q)} \langle FST(y), SND(y)\rangle : A \times B}}{x =_{\epsilon_\wedge(p,q)} y : A \times B}=_\eta}{f(x) =_{\mu_f(\epsilon_\wedge(p,q))} f(y) : A' \times B'}$$

$$\rhd_{mxc}$$

$$\dfrac{\dfrac{FST(x) =_p FST(y) : A}{g(FST(x)) =_{\mu_g(p)} g(FST(y)) : A'} \quad \dfrac{SND(x) =_q SND(y) : B}{h(SND(x)) =_{\mu_h(q)} h(SND(y)) : B'}}{\langle g(FST(x), h(SND(x))\rangle =_{\epsilon_\wedge(\mu_g(p),\mu_h(q))} \langle g(FST(y)), h(SND(y))\rangle : A' \times B'}$$
$$f(x) =_{\epsilon_\wedge(\mu_g(p),\mu_h(q))} f(y) : A' \times B'$$

And thus:

$$\mu_f(\epsilon_\wedge(p, q)) =_{mxc} \epsilon_\wedge(\mu_g(p), \mu_h(q))$$

■

4.6 Unit Type

For the unit type 1, our objective is to show the following theorem:

Theorem 4.20 *For any $x, y : 1$, there is a path t such that $x =_t y$. Moreover, $t = \rho$.*

Proof. To show that there is such t, we need to use the induction for the unit type [16]:

$$* \rhd_\eta x : 1$$

Therefore, given $x, y : 1$, we have:

$$\frac{x =_{\sigma(\eta)} * : 1 \qquad * =_\eta y : 1}{x =_{\tau(\sigma(\eta),\eta)} y : 1}$$

Moreover, by rule **4**, we have:

$$\tau(\sigma(\eta), \eta) =_{tsr} \rho.$$

Thus, $t \equiv \tau(\sigma(\eta), \eta)$ and $t =_{tsr} \rho$. ■

4.7 Univalence Axiom

The first thing to notice is that in our approach the following lemma holds:

Lemma 4.21 *For any types A and B, the following function exists:*

$$idtoeqv : (A = B) \to (A \simeq B)$$

Proof. The idea of the proof is similar to the one shown in [16]. We define *idtoeqv* to be $p_* : A \to B$. Thus, to end this proof, we just need to show that p_* is an equivalence.

For any path p, we can form a path $\sigma(p)$ and thus, we have $(\sigma(p))_* : B \to A$. Now, we show that $(\sigma(p)))_*$ is a quasi-inverse of p_*.

We need to check that:

1. $p_*((\sigma(p)_*(b)) = b$
2. $(\sigma(p))_*(p_*(a)) = a$

Both equations can be shown by an application of **Lemma 4.10**:

1. $p_*((\sigma(p)_*(b)) = (\sigma(p) \circ p)_*(b) = \tau(p, \sigma(p))_*(b) =_{tr} \rho_*(b) =_{\mu(p)} b.$
2. $(\sigma(p))_*(p_*(a)) = (p \circ \sigma(p))_*(a) = \tau(\sigma(p), p)_*(a) =_{tsr} \rho_*(a) =_{\mu(p)} a$

■

We have shown that a function exists, but we did not show that it is an equivalence. In fact, basic type theory cannot conclude that *idtoeqv* is an equivalence[16]. If we want this equivalence to be a property of our system, we must add a new axiom. This axiom is known as Voevodsky's univalence axiom[16]:

Axiom 4.22 *For any types A, B, idtoeqv is an equivalence, i.e., we have:*

$$(A = B) \simeq (A \simeq B)$$

4.8 Identity Type

In this section, we investigate specific lemmas and theorems related to the identity type. We start with the following theorem:

Theorem 4.23 *if $f : A \to B$ is an equivalence, then for $x, y : A$ we have:*

$$\mu_f : (x = y : A) \to (f(x) = f(y) : B)$$

Proof. We will omit the specific details of this proof, since it is equal to the one of **Theorem 2.11.1** presented in [16]. This is the case because this proof is independent of the usage of the induction principle of the identity type. The only difference is that at some steps we need to cancel inverse paths. In our approach, this is done by straightforward applications of **rules 3,4,5** and **6**. ■

Lemma 4.24 *For any $a : A$, with $x_1 =_p x_2$*

1. $transport^{x \to (a=x)}(p, q(x_1)) = \tau(q(x_1), p),$ *for* $q(x_1) : a = x_1$
2. $transport^{x \to (x=a)}(p, q(x_1)) = \tau(\sigma(p), q(x_1),$ *for* $q(x_1) : x_1 = a$
3. $transport^{x \to (x=x)}(p, q(x_1)) = \tau(\sigma(p), \tau(q(x_1), p))$ *for* $q(x_1) : x_1 = x_1$

Proof.

1. We start establishing the following reduction:

$$\frac{a =_{q(x_1)} x_1 \qquad x_1 =_p x_2}{a =_{\tau(q(x_1),p))} x_2} \rhd \quad a =_{q(x_2)} x_2$$

Thus, we just need to show that $transport^{x\to(a=x)}(p, q(x_1))$ also reduces to $a =_{q(x(2))} x_2$:

$$\frac{x_1 =_p x_2 \qquad q(x_1) : a = x_1}{p(x_1, x_2) \circ q(x_1) : a = x_2} =_{\mu(p)} \quad (a =_{q(x_2)} x_2)$$

2. We use the same idea:

$$\frac{x_2 =_{\sigma(p)} x_1 \qquad x_1 =_{q(x_1)} a}{x_2 =_{\tau(\sigma(p),q(x_1))} a} \rhd \quad x_2 =_{q(x_2)} a$$

$$\frac{x_1 =_p x_2 \qquad q(x_1) : x_1 = a}{p(x_1, x_2) \circ q(x_1) : x_2 = a} =_{\mu(p)} \quad (x_2 =_{q(x_2)} a)$$

3. Same as the previous cases:

$$\frac{\dfrac{x_2 =_{\sigma(p)} x_1 \qquad x_1 =_{q(x_1)} x_1}{x_2 =_{\tau(\sigma(p),q(x_1))} x_1} \qquad x_1 =_p x_2}{x_2 =_{\tau(\tau(\sigma(p),q(x_1)),p)} x_2} \rhd \quad x_2 =_{q(x_2)} x_2$$

$$\frac{x_1 =_p x_2 \qquad q(x_1) : x_1 = x_1}{p(x_1, x_2) \circ q(x_1) : x_2 = x_2} =_{\mu(p)} \quad (x_2 =_{q(x_2)} x_2)$$

■

4.9 Coproduct

One essential thing to remember is that a product $A + B$ has a left injection $inl : A \to A + B$ and $inr : B \to A + B$. As described in [16], it is expected that $A + B$ contains copies of A and B disjointly. In our path based approach, we achieve this by constructing every path $inl(a) = inl(b)$ and $inr(a) = inr(b)$ by applications of axiom μ on paths $a = b$. Thus we show that we get the following equivalences:

1. $(inl(a_1) = inl(a_2)) \simeq (a_1 = a_2)$
2. $(inr(b_1) = inr(b_2)) \simeq (b_1 = b_2)$
3. $(inl(a) = inr(b)) \simeq 0$

To prove this, we use the same idea as in [16]. We characterize the type:

$$(x \to (inl(a_0) = x)) : \Pi_{(x:A+B)}(inl(a_0 = x))$$

To do this, we define a type *code*:

$$x : A + B \vdash code(x) \text{ type}$$

Our main objective is to prove the equivalence $\Pi_{(x:A+B)}((inl(a_0) = x) \simeq code(x))$. Using the recursion principle of the coproduct, we can define *code* by two equations:

$$code(inl(a)) \equiv (a_0 = a)$$
$$code(inr(b)) \equiv 0$$

Theorem 4.25 *For any $x : A + B$, we have $inl(a_0 = x) \simeq code(x)$*

Proof. To show this equivalence, we use the same method as the one shown in [16]. The main idea is to define functions

$$encode : \Pi_{(x:A+B)}\Pi_{(p:inl(a_0)=x)}code(x)$$
$$decode : \Pi_{(x:A+B)}\Pi_{(c:code(x))}(inl(a_0) = x))$$

such that *decode* acts as a quasi-inverse of *encode*.
We start defining *encode*:

$$encode(x, s) \equiv transport^{code}(s, \rho_{a_0})$$

We notice that $\rho_{a_0} : code(inl(a_0))$, since $code(inl(a_0)) \equiv (a_0 =_\rho a_0)$ We also notice that for *encode*, it is only possible for the argument x to be of the form $x \equiv inl(a)$, since the other possibility is $x \equiv inr(a)$, but that case is not possible, because we would have a function to $code(inr(b)) \equiv 0$.

For *decode*, when $x \equiv inl(a)$, we have that $code(x) \equiv a_0 =_c a$ and thus, we define decode as $(inl(a_0) =_{\mu(c)} inl(a))$. When $x \equiv inr(a)$, then $code(x) \equiv 0$ and thus, we define *decode* as having any value, given by the elimination of the type 0. Now, we can finally prove the equivalence.

Starting with *encode*, we have $x \equiv inl(a)$, $inl(a_0) =_s x$. Since $encode(x, s) \equiv transport^{code}(s, \rho_{a_0})$, we have:

$$\dfrac{\dfrac{inl(a_0) =_s inl(a) \qquad \rho_{a_0} : code(inl(a_0))}{s(inl(a_0), inl(a)) \circ \rho_{a_0} : code(inl(a))}}{\rho_a : code(inl(a)) \equiv code(x)} =_{\mu(s)}$$

Now, we can go back to $inl(a_0) = inl(a)$ by an application of *decode*, since:

$$decode(\rho_a : code(x)) \equiv inl(a_0) =_{\mu_{inl}} inl(a)$$

And we conclude this part, since in our approach $inl(a_0) =_s inl(a)$ is constructed by applications of axiom μ.

Now, we start from decode. Let $c : code(x)$. If $x \equiv inl(a)$, then $c : a_0 = a$ and thus, $decode(c) \equiv inl(a_0) =_{\mu(c)} inl(a)$. Now, we apply *encode*. We have:

$$\begin{aligned} encode(x, \mu_c) &= transport^{code}(\mu_c, \rho_{a_0}) \\ &= transport^{a \to (a_0 = a)}(c, \rho_{a_0}) && \textbf{(Lemma 4.11)} \\ &= \tau(\rho_{a_0}, c) && \textbf{(Lemma 4.24)} \\ &= c && \textbf{(Rule 6)} \end{aligned}$$

If $x \equiv inr(b)$, we have that $c : 0$ and thus, as stated in [16], we can conclude anything we wish.

■

4.10 Reflexivity

In this section, our objective is to conclude an important result related to the reflexive path ρ:

Theorem 4.26 *For any type A and a path $x =_\rho x : A$, if a path s is obtained by a series (perhaps empty) of applications of axioms and rules of inference of $\lambda\beta\eta$-equality theory for type theory to the path ρ, then there is a path t' such that $s =_{t'} \rho$.*

Proof.

- Base Case:

  We can start only with a path $x =_\rho$. In that case, it is easy, since we have $\rho =_\rho \rho$.

Now, we consider the inductive steps. Starting from a path s and applying τ, σ, we already have rules yield the desired path:

- $s = \sigma(s')$, with $s' =_{t'} \rho$.

  In this case, we have $s = \sigma(s') = \sigma(\rho) =_{sr} \rho$.

- $s = \tau(s', s'')$, with $s' =_{t'} \rho$ and $s'' =_{t''} \rho$.

  We have that $s = \tau(s', s'') = \tau(\rho, \rho) =_{trr} \rho$

  The cases for applications of μ, ν and ξ remain to be proved. We introduce three new rules that handle these cases.

- $s = \mu(s')$, with $s' =_{t'} \rho$.

  We introduce rule **45**:

$$\frac{x =_{\rho_x} x : A \qquad [f : A \to B]}{f(x) =_{\mu(\rho_x)} f(x) : B(x)} \rhd_{mxp} \qquad f(x) =_{\rho_{f(x)}} f(x) : B(x)$$

  This rule is also valid for the dependent case:

$$\frac{x =_{\rho_x} x : A \qquad [f : \Pi_{(x:A)} B(x)]}{p(x, x) \circ f(x) =_{\mu(\rho_x)} f(x) : B(x)} \rhd_{mxp} \qquad f(x) =_{\rho_{f(x)}} f(x) : B(x)$$

  Thus, we have $s = \mu(s') = \mu(\rho) =_{mxp} \rho$.

- $s = \nu(s')$, with $s' =_{t'} \rho$.

  We introduce rule **46**:

$$\frac{f =_\rho f : \Pi_{(x:A)} B(x)}{f(x) =_{\nu(\rho_x)} f(x) : B(x)} \rhd_{nxp} \qquad f(x) =_{\rho_{f(x)}} f(x)$$

Thus, $s = \nu(s') = \nu(\rho) =_{nxp} \rho$.

- $s = \xi(s')$, with $s' =_{t'} \rho$.

  We introduce rule **47**:

$$\frac{b(x) =_\rho b(x) : B \qquad x : A}{\lambda x.b(x) =_{\xi(\rho)} \lambda x.b(x) : A \to B} \rhd_{xxp} \qquad \lambda x.b(x) =_\rho \lambda x.b(x)$$

  Thus, $s = \xi(s') = \xi(\rho) =_{xxp} \rho$.

■

4.11 Natural Numbers

The Natural Numbers is a type defined inductively by an element $0 : \mathbb{N}$ and a function $succ : \mathbb{N} \to \mathbb{N}$. In our approach, the path space of the naturals is also characterized inductively. We start from the reflexive path $0 =_\rho 0$. All subsequent paths are constructed by applications of the inference rules of $\lambda\beta\eta$-equality. We show that this characterization is similar to the one constructed in [16]. To do this, we use *code*, *encode* and *decode*. For $\mathbb{N}$, we define *code* recursively [16]:

$$\begin{gathered} code(0,0) \equiv 1 \\ code(succ(m),0) \equiv 0 \\ code(0,succ(m)) \equiv 0 \\ code(succ(m),succ(n)) \equiv code(m,n) \end{gathered}$$

We also define a dependent function $r : \Pi_{(n:\mathbb{N})} code(m,n)$, with:

$$\begin{gathered} r(0) \equiv * \\ r(succ(n)) \equiv r(n) \end{gathered}$$

Theorem 4.27 *For any $m, n : \mathbb{N}$, if there is a path $m =_t n : \mathbb{N}$, then $t \rhd \rho$.*

Proof. Since all paths are constructed from the reflexive path $0 =_\rho 0$, this is a direct application of **Theorem 4.26**. ■

Theorem 4.28 *For any $m, n : \mathbb{N}$, we have $(m = n) \simeq code(m,n)$*

Proof. We need to define *encode* and *decode* and prove that they are quasi-inverses. We define $encode : \Pi_{(m,n:\mathbb{N})}(m = n) \to code(m, n)$ as:

$$encode(m, n, p) \equiv transport^{code(m,-)}(p, r(m))$$

We define $decode : \Pi_{(m,n:\mathbb{N})} code(m, n) \to (m = n)$ recursively:

$$\begin{gathered} decode(0, 0, c) \equiv 0 =_\rho 0 \\ decode(succ(m), 0, c) \equiv 0 \\ decode(0, succ(m), c) \equiv 0) \\ decode(succ(m), succ(n), c) \equiv \mu_{succ}(decode(m, n, c)) \end{gathered}$$

We now prove that if $m =_p n$, then $decode(code(m, n)) = \rho$. We prove by induction. The base is trivial, since $decode(0, 0, c) \equiv \rho$. Now, consider $decode(succ(m), succ(n), c)$. We have that

$$decode(succ(m), succ(n), c) \equiv \mu_{succ}(decode(m, n, c))$$

By the inductive hypothesis, $decode(m, n, c) \equiv \rho$. Thus, we need to prove that $\mu_{succ} = \rho$. This last step is a straightforward application of **rule 47**. Therefore, $\mu_{succ} =_{mxp} \rho$. With this information, we can start the proof of the equivalence.

For any $m =_p n$, we have:

$$encode(m, n, p) \equiv transport^{code(m,-)}(p, r(m))$$

Thus:

$$\frac{m =_p n \qquad r(m) : code(m, m)}{p(m, n) \circ r(m) : code(m, n)} =_{\mu(p)} \quad (r(n) : code(m, n))$$

Now, we know that $decode(r(n) : code(m, n)) = \rho$ and,by **Theorem 4.27**, $p = \rho$.

The proof starting from a $c : code(m, n)$ is equal to the one presented in [16]. We prove by induction. If m and n are 0, we have the trivial path $0 =_\rho 0$, thus $decode(0, 0, c) = \rho_0$, whereas $encode(0, 0, \rho_0) \equiv r(0) \equiv *$. We conclude this part recalling that every $x : 1$ is equal to $*$, since we have $x =_{\sigma(\eta)} * : 1$. In the case of $decode(succ(m), 0, c)$ or $decode(0, succ(n), c)$, $c : 0$. The only case left is for $decode(succ(m), succ(n), c)$. Similar to [16], we prove by induction:

$$\begin{aligned} &encode(succ(m), succ(n), decode(succ(m), succ(n), c)) \\ &\qquad = encode(succ(m), succ(n), \mu_{succ}(decode(m, n, c)) \end{aligned}$$

$$= transport^{code(succ(m),-)}(\mu_{succ}(decode(m,n,c)), r(succ(m))$$
$$= transport^{code(succ(m),succ(-)}(decode(m,n,c), r(succ(m)))$$
$$= transport^{code(m,-)}(decode(m,n,c), r(m))$$
$$= encode(m,n,decode(m,n,c))$$
$$= c$$ ■

4.12 Fundamental Group of a Circle

The objective of this section is to show that it is possible to use computational paths to obtain one of the main results of homotopy theory, the fact that the fundamental group of a circle is isomorphic to the integers group. First, we define a circle as follows:

Definition 4.29 (The circle S^1) *A circle is the type generated by:*

- *A point* $base : S^1$
- *A computational path* $base =_{loop} base : S^1$.

The first thing one should notice is that this definition doest not use only the points of the type S^1, but also a computational path *loop* between those points. That is way it is called a higher inductive type [16]. Our approach differs from the classic one on the fact that we do not need to simulate the path-space between those points, since computational paths exist in the syntax of the theory. Thus, if one starts with a path $base =_{loop} base : S^1$., one can naturally obtain additional paths applying the path-axioms ρ, τ and σ. Thus, one has a path $\sigma(loop) = loop^{-1}$, $\tau(loop, loop)$, etc. In classic type theory, the existence of those additional paths comes from establishing that the paths should be freely generated by the constructors [16]. In our approach, we do not have to appeal for this kind of argument, since all paths comes naturally from direct applications of the axioms.

With that in mind, one can define the fundamental group of a circle. In homotopy theory, the fundamental group is the one formed by all equivalence classes up to homotopy of paths (loops) starting from a point a and also ending at a. Since the we use computational paths as the syntax counterpart in type theory of homotopic paths, we use it to propose the following definition:

Definition 4.30 ($\Pi_1(A, a)$ structure) *$\Pi_1(A, a)$ is a structure defined as follows:*

$$\Pi_1(A, a) = \{[path]_{rw} \mid a =_{path} a : A\}$$

We use this structure to define the fundamental group of a circle. We also need to show that it is indeed a group.

Proposition 4.31 $(\Pi_1(S,a),\circ)$ *is a group.*

Proof. The first thing to define is the group operation $\circ$. Given any $a =_r a : S^1$ and $a =_t a : S^1$, we define $r \circ s$ as $\tau(s,r)$. Thus, we now need to check the group conditions:

- Closure: Given $a =_r a : S^1$ and $a =_t a : S^1$, $r \circ s$ must be a member of the group. Indeed, $r \circ s = \tau(s,r)$ is a computational path $a =_{\tau(s,r)} a : S^1$.

- Inverse: Every member of the group must have an inverse. Indeed, if we have a path r, we can apply $\sigma(r)$. We claim that $\sigma(r)$ is the inverse of r, since we have:

$$\sigma(r) \circ r = \tau(r, \sigma(r)) =_{tr} \rho$$
$$r \circ \sigma(r) = \tau(\sigma(r), r) =_{tsr} \rho$$

  Since we are working up to rw-equality, the equalities hold strictly.

- Identity: We use the path $a =_\rho a : S^1$ as the identity. Indeed, we have:

$$r \circ \rho = \tau(\rho, r) =_{tlr} r$$
$$\rho \circ r = \tau(r, \rho) =_{trr} r.$$

- Associativity: Given any members of the group $a =_r a : S^1$, $a =_t a$ and $a =_s a$, we want that $r \circ (s \circ t) = (r \circ s) \circ t$:

$$r \circ (s \circ t) = \tau(\tau(t,s), r) =_{tt} \tau(t, \tau(s,r)) = (r \circ s) \circ t$$

All conditions have been satisfied. $(\Pi_1(S,a),\circ)$ is a group. ■

Thus, $(\Pi_1(S,a),\circ)$ is indeed a group. We call this group the fundamental group of S^1. Therefore, the objective of this section is to show that $\Pi_1(S,a) \cong \mathbb{Z}$.

Before we start developing this proof, the following lemma will prove to be useful:

Lemma 4.32 *All paths generated by a path $a =_{loop} a$ are rw-equal to a path $loop^n$, for a $n \in \mathbb{Z}$.*

We have said that from a *loop*, one freely generate different paths applying the composition τ and the symmetry. Thus, one can, for example, obtain something such as $loop \circ loop \circ loop^{-1} \circ loop$.... Our objective with this lemma is to show that, in fact, this path can be reduced to a path of the form $loop^n$, for $n \in \mathbb{Z}$.

Proof. The idea is to proceed by induction on the number n of loops, i.e., $loop^n$. We start from a base ρ. For the base case, it is trivially true, since we define it to be equal to $loop^0$. From ρ, one can construct more complex paths by composing with *loop* or $\sigma(loop)$ on each step. We have the following induction steps:

- A path of the form ρ concatenated with *loop*: We have $\rho \circ loop = \tau(loop, \rho) =_{trr} loop = loop^1$;

- A path of the form ρ concatenated with $\sigma(loop)$: We have $\rho \circ \sigma(loop) = \tau(\sigma(loop), \rho) =_{trr} = \sigma(loop) = loop^{-1}$

- A path of the form $loop^n$ concatenated with *loop*: We have $loop^n \circ loop = loop^{n+1}$.

- A path of the form $loop^n$ concatenated with $\sigma(loop)$: We have $loop^n \circ \sigma(loop) = (loop^{n-1} \circ loop) \circ \sigma(loop) =_{tt} loop^{n-1} \circ (loop \circ \sigma(loop)) = loop^{n-1} \circ (\tau(\sigma(loop), loop)) =_{tsr} = loop^{n-1} \circ \rho = \tau(\rho, loop^{n-1}) =_{tlr} loop^{n-1}$

- A path of the form $loop^{-n}$ concatenated with *loop*: We have $loop^{-n} = loop^{-(n-1)} \circ loop^{-1} = loop^{-(n-1)} \circ \sigma(loop)$. Thus, we have $(loop^{-(n-1)} \circ \sigma(loop)) \circ loop =_{tt} loop^{-(n-1)} \circ (\sigma(loop) \circ loop) = loop^{-(n-1)} \circ \tau(loop, \sigma(loop)) =_{tr} = loop^{-(n-1)} \circ \rho = \tau(\rho, loop^{-(n-1)}) =_{tlr} loop^{-(n-1)}$.

- a path of the form $loop^{-n}$ concatenated with $\sigma(loop)$: We have $loop^{-n} \circ loop^{-1} = loop^{-(n+1)}$

Thus, every path is of the form $loop^n$, with $n \in \mathbb{Z}$. ■

This lemma shows that every path of the fundamental group can be represented by a path of the form $loop^n$, with $n \in \mathbb{Z}$.

Theorem 4.33 $\Pi_1(S, a) \cong \mathbb{Z}$

To prove this theorem, one could use the approach proposed in [16], defining an encode and decode functions. Nevertheless, since our computational paths are part of the syntax, one does not need to rely on this kind of approach to simulate a path-space, we can work directly with the concept of path.

Theorem 4.34 $\Pi_1(S, a) \cong \mathbb{Z}$

To prove this theorem, one could use the approach proposed in [16], defining an encode and decode functions. Nevertheless, since our computational paths are part of the syntax, one does not need to rely on this kind of approach to simulate a path-space, we can work directly with the concept of path.

Proof. The proof is done by establishing a function from $\Pi_1(S, a)$ to $\mathbb{Z}$ and then an inverse from $\mathbb{Z}$ to $\Pi_1(S, a)$. Since we have access to the previous lemma, this task is not too difficult. The main idea is that the n on $loop^n$ means the amount of times one goes around the circle, while the sign gives the direction (clockwise or anti-clockwise). In other words, it is the *winding* number. Since we have shown that every path of the fundamental group is of the form $loop^n$, with $n \in \mathbb{Z}$, then we just need to translate $loop^n$ to an integer n and an integer n to a path $loop^n$. We define two functions, $toInteger : \Pi_1(S, a) \to \mathbb{Z}$ and $toPath : \mathbb{Z} \to \Pi_1(S, a)$:

- *toInteger*: To define this function, we use the help of two functions defined in $\mathbb{Z}$: the successor function *succ* and the predecessor function *pred*. We define *toInteger* as follows. Of course, we use directly the fact that every loop of S is of the form $loop^n$ with $n \in \mathbb{Z}$:

$$toInteger : \begin{cases} toInteger([loop^n]_{rw} \equiv [\rho]_{rw}) = 0 & n = 0 \\ toInteger([loop^n]_{rw}) = succ(toInteger([loop^{n-1}]_{rw})) & n > 0 \\ toInteger([loop^n]_{rw}) = pred(toInteger([loop^{n+1}])]_{rw}) & n < 0 \end{cases}$$

- *toPath*: We just need to transform an integer n into a path $loop^n$:

$$toPath : \begin{cases} toPath(n) = [\rho]_{rw} & n = 0 \\ toPath(n) = toPath(n-1) \circ [loop]_{rw} & n > 0 \\ toPath(n) = toPath(n+1) \circ [\sigma(loop)]_{rw} & n < 0 \end{cases}$$

Now we just need to show that they are inverses. To do this, we have to check two equations:

1. $toPath(toInteger([x]_{rw})) = [x]_{rw}$
2. $toInteger(toPath(n)) = n$

From a path $[x]_{rw}$, we apply Lemma 5.22 to obtain $[x]_{rw} = [loop^n]_{rw}$ for some integer n. Thus $toPath(toInteger([x]_{rw})) = toPath(toInteger([loop^n]_{rw})) = toPath(n) = [loop^n]_{rw} = [x]_{rw}$. The opposite direction is straightforward: $toInteger(toPath(n)) = toInteger([loop^n]_{rw}) = n$. Thus, we conclude that they are inverses. Also, the map between the operations of the groups is direct, since we can easily map $\circ$ to $+$. This is due to the fact that $loop^n \circ loop^m = loop^{n+m}$ and thus, $toInteger[[loop^{n+m}]_{rw} = n + m$. It is also straightforward that $toPath(n + m) = [loop^{n+m}]_{rw}$. Thus, we establish the isomorphism $\Pi_1(S, a) \cong \mathbb{Z}$. ■

4.13 Rules Added to $LND_{EQ} - TRS$

In this section, we have introduced 7 new rules to the $LND_{EQ} - TRS$ system. It is the following list of rules:

40. $\tau(\mu(r), \mu(s)) =_{tf} \mu(\tau(r, s))$
41. $\mu_g(\mu_f(p)) =_{cf} \mu_{g \circ f}(p)$
42. $\mu_{Id_A}(p) =_{ci} p$
43. $\tau(H_{f,g}(x), \mu_g(p)) =_{hp} \tau(\mu_f(p), H_{f,g}(y))$
44. $\mu_f(\epsilon_\wedge(p, q)) =_{mxc} \epsilon_\wedge(\mu_g(p), \mu_h(q))$
45. $\mu_f(\rho_x) =_{mxp} \rho_{f(x)}$
46. $\nu(\rho_x) =_{nxp} \rho_{f(x)}$
47. $\xi(\rho) =_{xxp} \rho$

Since we added 8 new rules to the system, a rather natural question may arise: what about the completeness of this rewrite system. Is it now complete? The formal answer to this question is that we have not proved the completeness of this system yet. In this work we added rules that were not previously found and that involve simple combinations of computations of inference rules such as μ and ν. We then showed that our system is now powerful enough to formalize all basic types of homotopy type theory. Nevertheless, we think that it is important to formalize a proof of completeness of our rewrite system and this will be one of the main topics of research for future works.

Another important topic is the proof of termination and confluence of this system. We have mentioned that this proof was obtained in the works of [2, 4, 3, 6]. Nevertheless, since we added new rules, we need to extend this proof to include these rules. To do that, we can use the same techniques

used in these previous works. Even so, extending this proof might not be completely straightforward and could involve a good amount of work. Given the importance of this proof, we also leave this work as one of the main focus of our future works.

5 Conclusion

In this work, we connected our computational path approach to homotopy type theory. Using the algebra of computational paths, we have established important results of homotopy type theory. That way, we have shown that our approach yields the main building blocks of homotopy type theory, on par with the classic approach. We have also improved the rewrite system, adding new reduction rules. Indeed, we have ended this work with one of the most classic results of algebraic topology, the fact that the fundamental group of the circle is isomorphic to the group of integers.

In view of all results achieved in this work, we have developed a valid alternative approach to the identity type and homotopy type theory, based on this algebra of paths. We also believe that we have opened the way, in future works, for possible expansions of this results, opening the possibility of formulating and proving even more intricate concepts and theorems of homotopy type theory using computational paths. We have also used the results obtained in section 4 to calculate the fundamental group of other spaces besides the circle. We present these calculations in a new and still unpublished work, but already available in a preprint version that can be checked in [9].

References

[1] Steve Awodey. Type theory and homotopy. In P. Dybjer, Sten Lindström, Erik Palmgren, and G. Sundholm, editors, *Epistemology versus Ontology*, volume 27 of *Logic, Epistemology, and the Unity of Science*, pages 183–201. Springer Netherlands, 2012.

[2] A. G. de Oliveira. *Proof transformations for labelled natural deduction via term rewriting*. 1995. Master's thesis, Depto. de Informática, Universidade Federal de Pernambuco, Recife, Brazil, April 1995.

[3] A. G. de Oliveira and R. J. G. B. de Queiroz. A normalization procedure for the equational fragment of labelled natural deduction. *Logic Journal of IGPL*, 7(2):173–215, 1999.

[4] R. J. G. B. de Queiroz and A. G. de Oliveira. Term rewriting systems with labelled deductive systems. In *Proceedings of Brazilian Symposium on Artificial Intelligence (SBIA'94)*, pages 59–72, 1994.

[5] R. J. G. B. de Queiroz and A. G. de Oliveira. Natural deduction for equality: The missing entity. In Luiz Carlos Pereira, Edward Haeusler, and Valeria de Paiva, editors, *Advances in Natural Deduction - A Celebration of Dag Prawitz's Work*, pages 63–91. Springer, 2014.

[6] R. J. G. B. de Queiroz, A. G. de Oliveira, and D. M. Gabbay. *The Functional Interpretation of Logical Deduction*. World Scientific, 2011.

[7] R. J. G. B. de Queiroz, A. G. de Oliveira, and A. F. Ramos. Propositional equality, identity types, and direct computational paths. *South American Journal of Logic*, 2(2):245–296, 2016. Special Issue A Festschrift for Francisco Miraglia, M. E. Coniglio, H. L. Mariano and V. C. Lopes (Guest Editors).

[8] R. J. G. B. de Queiroz and D. M. Gabbay. Equality in labelled deductive systems and the functional interpretation of propositional equality. In *Proceedings of the 9th Amsterdam Colloquium*, pages 547–565. ILLC/Department of Philosophy, University of Amsterdam, 1994.

[9] Tiago M. L. De Veras, Arthur F. Ramos, Ruy J. G. B. De Queiroz, and Anjolina G. De Oliveira. On the calculation of fundamental groups in homotopy type theory by means of computational paths. *https://arxiv.org/abs/1804.01413*, 2018.

[10] R. Harper. Type theory foundations, 2012. Type Theory Foundations, Lecture at Oregon Programming Languages Summer School, Eugene, Oregon.

[11] J. Roger Hindley and Jonathan P. Seldin. *Lambda-calculus and combinators: an introduction*. Cambridge University Press, 2008.

[12] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 83–111. Oxford Univ. Press, New York, 1998.

[13] Philippe Le Chenadec. On the logic of unification. *Journal of Symbolic computation*, 8(1):141–199, 1989.

[14] Arthur F. Ramos, R. J. G. B. de Queiroz, and A. G. de Oliveira. On the identity type as the type of computational paths. 2015. http://arxiv.org/abs/1504.04759.

[15] Arthur F. Ramos, Ruy J. G. B. De Queiroz, and Anjolina G. De Oliveira. On the identity type as the type of computational paths. *Logic Journal of the IGPL*, 25(4):562–584, 2017.

[16] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

[17] V. Voevodsky. Univalent foundations and set theory, 2014. Univalent Foundations and Set Theory, Lecture at IAS, Princeton, New Jersey, Mar 2014.

Arthur F. Ramos
Centro de Informática
Universidade Federal de Pernambuco
Av. Jornalista Anibal Fernandes, s/n
50740-560 Recife, PE, Brazil
E-mail: afr@cin.ufpe.br

Ruy J. G. B. de Queiroz
Centro de Informática
Universidade Federal de Pernambuco
Av. Jornalista Anibal Fernandes, s/n
50740-560 Recife, PE, Brazil
E-mail: ruy@cin.ufpe

Anjolina G. de Oliveira
Centro de Informática
Universidade Federal de Pernambuco
Av. Jornalista Anibal Fernandes, s/n
50740-560 Recife, PE, Brazil
E-mail: ago@cin.ufpe.br

Tiago M. L. de Veras
Centro de Informática
Universidade Federal de Pernambuco
Av. Jornalista Anibal Fernandes, s/n
50740-560 Recife, PE, Brazil
E-mail: tiago.veras@ufrpe.br

Appendices

A Subterm Substitution

In Equational Logic, the sub-term substitution is given by the following inference rule [4]:

$$\frac{s = t}{s\theta = t\theta}$$

One problem is that such rule does not respect the sub-formula property. To deal with that, [13] proposes two inference rules:

$$\frac{M = N \qquad C[N] = O}{C[M] = O}\, IL \qquad \frac{M = C[N] \qquad N = O}{M = C[O]}\, IR$$

where M, N and O are terms.

As proposed in [7], we can define similar rules using computational paths, as follows:

$$\frac{x =_r \mathcal{C}[y] : A \qquad y =_s u : A'}{x =_{\mathtt{sub_L}(r,s)} \mathcal{C}[u] : A} \qquad \frac{x =_r w : A' \qquad \mathcal{C}[w] =_s u : A}{\mathcal{C}[x] =_{\mathtt{sub_R}(r,s)} u : A}$$

where C is the context in which the sub-term detached by '[ ]' appears and A' could be a sub-domain of A, equal to A or disjoint to A.

In the rule above, $\mathcal{C}[u]$ should be understood as the result of replacing every occurrence of y by u in C.

B List of Rewrite Rules

We present the rewrite rules of $LND_{EQ} - TRS$. They are as follows (We show only the original 39 rules as proposed by [2] and [7]. The new rules added to the system appears in the end of section 5):

1. $\sigma(\rho) \rhd_{sr} \rho$
2. $\sigma(\sigma(r)) \rhd_{ss} r$
3. $\tau(\mathcal{C}[r], \mathcal{C}[\sigma(r)]) \rhd_{tr} \mathcal{C}[\rho]$
4. $\tau(\mathcal{C}[\sigma(r)], \mathcal{C}[r]) \rhd_{tsr} \mathcal{C}[\rho]$
5. $\tau(\mathcal{C}[r], \mathcal{C}[\rho]) \rhd_{trr} \mathcal{C}[r]$
6. $\tau(\mathcal{C}[\rho], \mathcal{C}[r]) \rhd_{tlr} \mathcal{C}[r]$
7. $\mathtt{sub_L}(\mathcal{C}[r], \mathcal{C}[\rho]) \rhd_{slr} \mathcal{C}[r]$
8. $\mathtt{sub_R}(\mathcal{C}[\rho], \mathcal{C}[r]) \rhd_{srr} \mathcal{C}[r]$
9. $\mathtt{sub_L}(\mathtt{sub_L}(s, \mathcal{C}[r]), \mathcal{C}[\sigma(r)]) \rhd_{sls} s$

10. $\mathtt{sub}_{\mathtt{L}}(\mathtt{sub}_{\mathtt{L}}(s, \mathcal{C}[\sigma(r)]), \mathcal{C}[r]) \rhd_{slss} s$
11. $\mathtt{sub}_{\mathtt{R}}(\mathcal{C}[s], \mathtt{sub}_{\mathtt{R}}(\mathcal{C}[\sigma(s)], r)) \rhd_{srs} r$
12. $\mathtt{sub}_{\mathtt{R}}(\mathcal{C}[\sigma(s)], \mathtt{sub}_{\mathtt{R}}(\mathcal{C}[s], r)) \rhd_{srrr} r$
13. $\mu_1(\xi_1(r)) \rhd_{mx2l1} r$
14. $\mu_1(\xi_\wedge(r, s)) \rhd_{mx2l2} r$
15. $\mu_2(\xi_\wedge(r, s)) \rhd_{mx2r1} s$
16. $\mu_2(\xi_2(s)) \rhd_{mx2r2} s$
17. $\mu(\xi_1(r), s, u) \rhd_{mx3l} s$
18. $\mu(\xi_2(r), s, u) \rhd_{mx3r} u$
19. $\nu(\xi(r)) \rhd_{mxl} r$
20. $\mu(\xi_2(r), s) \rhd_{mxr} s$
21. $\xi(\mu_1(r), \mu_2(r)) \rhd_{mx} r$
22. $\mu(t, \xi_1(r), \xi_2(s)) \rhd_{mxx} t$
23. $\xi(\nu(r)) \rhd_{xmr} r$
24. $\mu(s, \xi_2(r)) \rhd_{mx1r} s$
25. $\sigma(\tau(r, s)) \rhd_{stss} \tau(\sigma(s), \sigma(r))$
26. $\sigma(\mathtt{sub}_{\mathtt{L}}(r, s)) \rhd_{ssbl} \mathtt{sub}_{\mathtt{R}}(\sigma(s), \sigma(r))$
27. $\sigma(\mathtt{sub}_{\mathtt{R}}(r, s)) \rhd_{ssbr} \mathtt{sub}_{\mathtt{L}}(\sigma(s), \sigma(r))$
28. $\sigma(\xi(r)) \rhd_{sx} \xi(\sigma(r))$
29. $\sigma(\xi(s, r)) \rhd_{sxss} \xi(\sigma(s), \sigma(r))$
30. $\sigma(\mu(r)) \rhd_{sm} \mu(\sigma(r))$
31. $\sigma(\mu(s, r)) \rhd_{smss} \mu(\sigma(s), \sigma(r))$
32. $\sigma(\mu(r, u, v)) \rhd_{smsss} \mu(\sigma(r), \sigma(u), \sigma(v))$
33. $\tau(r, \mathtt{sub}_{\mathtt{L}}(\rho, s)) \rhd_{tsbll} \mathtt{sub}_{\mathtt{L}}(r, s)$
34. $\tau(r, \mathtt{sub}_{\mathtt{R}}(s, \rho)) \rhd_{tsbrl} \mathtt{sub}_{\mathtt{L}}(r, s)$
35. $\tau(\mathtt{sub}_{\mathtt{L}}(r, s), t) \rhd_{tsblr} \tau(r, \mathtt{sub}_{\mathtt{R}}(s, t))$
36. $\tau(\mathtt{sub}_{\mathtt{R}}(s, t), u) \rhd_{tsbrr} \mathtt{sub}_{\mathtt{R}}(s, \tau(t, u))$
37. $\tau(\tau(t, r), s) \rhd_{tt} \tau(t, \tau(r, s))$
38. $\tau(\mathcal{C}[u], \tau(\mathcal{C}[\sigma(u)], v)) \rhd_{tts} v$
39. $\tau(\mathcal{C}[\sigma(u)], \tau(\mathcal{C}[u], v)) \rhd_{tst} u.$