A TOPOLOGICAL APPLICATION OF LABELLED NATURAL DEDUCTION

A PREPRINT

Tiago Mendonça Lucena de Veras
Departamento de Matemática
Universidade Federal Rural de Pernambuco
Recife, Brasil
tiago.veras@ufrpe.br

Arthur F. Ramos
Microsoft, Redmond, USA
arfreita@microsoft.com

Ruy J. G. B. de Queiroz
Centro de Informática
Universidade Federal de Pernambuco
Recife, Brasil
ruy@cin.ufpe.br

Anjolina G. de Oliveira
Centro de Informática
Universidade Federal de Pernambuco
Recife, Brasil
ago@cin.ufpe.br

May 11, 2021

ABSTRACT

Using a natural deduction system in the Curry–Howard tradition, we demonstrate how to formalise the concept of computational paths (sequences of rewrites) as equalities between two terms of the same type. The aim is to formulate a term rewriting system in order to illustrate how one can perform computations within these computational paths, establishing equalities between equalities. We shall proceed to use the Labelled Natural Deduction – LND – based on the concept of computational paths (which is a system of rewrites) as a tool for obtaining the results on the fundamental group of the circle, the torus and the real projective plane.

Keywords Fundamental Group · Labelled Natural Deduction · Term Rewriting System · Computational Paths · Algebraic Topology.

1 Introduction

The identity type is arguably one of the most interesting entities of Martin-Löf type theory. From any type $A$, it is possible to construct the identity type $Id_A(x, y)$. This type establishes the relation of identity between two terms of $A$, i.e., if there is $x =_p y : A$, then $p$ is a witness or proof that $x$ is indeed equal to $y$. The proposal of the Univalence Axiom made the identity type one of the most studied aspects of type theory. It proposes that $x = y$ is equivalent to saying that $x \simeq y$, that is, the identity is equivalent to equivalence. Another important aspect is the fact that it is possible to interpret the identity type as paths between two points of the same space. This interpretation gives rise to the interesting rendition of equality as a collection of homotopical paths. This connection of type theory and homotopy theory makes type theory a suitable foundation for both computation and mathematics. Nevertheless, this interpretation is only a semantical one and it has not been proposed with a syntactical counterpart for the concept of path in type theory (cf. Voevodsky’s observation\textsuperscript{1}). For that reason, the addition of paths to the syntax of homotopy type theory has been recently proposed by de Queiroz, Ramos and de Oliveira \textsuperscript{[27][32]}. The idea is to use an entity named computational

\textsuperscript{1}As it turns out, in the formulation of the identity type, there is only one introduction rule, and thus only one identifier/constructor, i.e. ‘refl’, to be used to construct elements of $Id_A(a, b)$. Even though the ‘transport’ constructor allows for the construction of other paths in the syntax, the fact that the only base case is ‘refl’ has brought some mystery as to whether terms for paths are actually integrated in the whole conceptual and formal framework of type theory. From an email message by M. Escardò:
path, proposed by de Queiroz and Gabbay in 1994 \cite{24}, and demonstrate that it can be useful for formalizing the identity type in a more explicit manner.

On the other hand, one of the main interesting points of the interpretation of logical connectives via deductive systems which use a labelling system is the clear separation between a functional calculus on the labels (the names that record the steps of the proof) and a logical calculus on the formulas \cite{15,24}. Moreover, this interpretation has important applications. The works of \cite{27,24,29,31} claim that the harmony that comes with this separation makes labelled natural deduction a suitable framework to study and develop a theory of equality for natural deduction. Take, for example, the following cases taken from the \(\lambda\)-calculus \cite{29}:

\[
(\lambda x. (\lambda y. yx)(\lambda w. zw))v \xrightarrow{\eta} (\lambda x. (\lambda y. yx)z)v \xrightarrow{\beta} (\lambda y. yv)z \xrightarrow{\beta} vz \\
(\lambda x. (\lambda y. yx)(\lambda w. zw))v \xrightarrow{\beta} (\lambda x(\lambda w.zw)x)v \xrightarrow{\eta} (\lambda x.zx)v \xrightarrow{\beta} vz
\]

In the theory of the \(\beta\eta\)-equality of \(\lambda\)-calculus, we can indeed say that \((\lambda x. (\lambda y. yx)(\lambda w. zw))v\) is equal to \(zv\). Moreover, as can be seen above, there are at least two ways of obtaining these equalities. We can go further, and call \(s\) the first sequence of \textit{rewrites} that establish that \((\lambda x. (\lambda y. yx)(\lambda w. zw))v\) is indeed equal to \(zv\). The second one, for example, can be called \(r\). Thus, we can say that this equality is established by \(s\) and \(r\). As we shall see in this paper, \(s\) and \(r\) are examples of an entity known as \textit{computational path}.

Since we now have labels (computational paths) that establish the equality between two terms, interesting questions might arise: (i) is \(s\) different from \(r\); (ii) are there normal forms of this equality proof; (iii) if \(s\) is equal to \(r\), how can we prove this? Questions like these can be answered when we work in a labelled natural deduction framework. The idea is that we are not limited by the calculus on the formulas, but we can also define and work with rules that apply to the labels. That way, we can use these rules to formally establish the equality between these labels, i.e., establish equalities between equalities. Here we shall use a system proposed by \cite{21} and known as \textit{LND\(\text{EQ} - \text{TRS}\)}.

In the vein of bringing to public record things that Vladimir said, here is a short interview.

--- Forwarded Message ---

Subject: Re: historical question
Date: Thu, 22 Oct 2015 16:08:14 -0400
From: Vladimir Voevodsky <vlad...@ias.edu>
To: Martin Escardo <m.es...@cs.bham.ac.uk>
CC: Prof. Vladimir Voevodsky <vlad...@ias.edu>

(...) 

"> (Also: what was your first reaction when you saw the identity type for > the first time? Did you immediately connect it with path spaces?)

Not at all. I did not make this connection until late 2009. All the time before it I was hypnotized by the mantra that the only inhabitant of the Id type is reflexivity which made then [sic] useless from my point of view.”

Vladimir.

published in a google discussion group of Homotopy Type Theory on Thursday, 12 October 2017 20:24:26, with Vladimir Voevodsky in 2015. https://groups.google.com/g/homotopytypetheory/c/K_4bAZEDRvE/m/VbYjok7bAAAJ

2By introducing terms as \textit{explicit} conversion paths between terms, allowing for further iterations coming from conversions between those paths, we make an attempt at dealing with an important issue brought about by Voevodsky, where by defining a rule stating that ‘any two terms of an equality type are equivalent under conversions’ one would immediately resurrect the undecidability of the identity type:

“The introduction of the equality types however brings with it a whole new dimension to the type system because the type expressions \(e_{R\Theta}(r, r')\) unlike the type expressions we have considered before depend on terms. Another issue which arises is how to ensure that the equality types are mapped by models to \(pt\) or \(\emptyset\) and not to sets with many elements. One can impose the later condition by a rule saying that any two terms of an equality type are equivalent under conversions but this immediately resurrects the un-decidability argument. I do not know of any satisfactory solution of these problems in the usual type theories.” \cite{43} (p.23)

3The formalization of such an iteration between equalities seems to find a counterpart in Voevodsky’s observations on the ‘isomorphism invariance principle’:

“One of the keystones of contemporary mathematics is the \textbf{isomorphism invariance principle}: for any statement \(P\) about \(X\) and any isomorphism \(X \xrightarrow{\sim} X'\), there is a statement \(P_0\) about \(X'\) such that \(P\) holds iff \(P_0\) holds. The equality problem in formalizations comes in part from the fact that when one encodes \(X\) and \(X'\) the isomorphism is lost. There is more to the equality problem than isomorphism invariance:
In that context, the contribution of this paper will be to propose an interesting connection: it is possible to use a labelled natural deduction system together with $LND_{EQ} - TRS$ to obtain topological results about fundamental groups.

Indeed, in this paper we will develop a framework for dealing with such expressions representing computational paths and show that it is powerful enough to provide the formal tools to calculate the fundamental group of the circle, the torus and the real projective plane. For this, we use a labelled deduction system based on the concept of computational paths (sequence of rewrites). Taking into account that in mathematics the calculation of this fundamental group is quite laborious, we believe our work accomplishes this calculation in a less complex form. Nevertheless, to obtain this result we first need to formally define the concept of computational paths and define $LND_{EQ} - TRS$.

As a matter of fact, this is part of an ongoing project which, while it looks for the utility and the impact of the so-called Curry-Howard interpretation of logical deduction in the actual practice of an important area of mathematics, namely homotopy theory. The short citation for the Royal Swedish Academy of Sciences’ “2020 Rolf Schock Prize in logic and philosophy” says that it was awarded to Per Martin-Löf (shared with Dag Prawitz) “for the creation of constructive type theory.” In a longer statement, the prize committee recalls that constructive type theory is “a formal language in which it is possible to express constructive mathematics” (…) “[which] also functions as a powerful programming language and has had an enormous impact in logic, computer science and, recently, mathematics.”

In fact, by introducing a framework whose formalisation of the logical notion of equality is done via the so-called “identity type”, we have the possibility for a surprising connection between term rewriting and geometric concepts such as groupoids in the study of semantics of computation, it also seeks to demonstrate the utility and the impact of the so-called Curry-Howard interpretation of logical deduction in the actual practice of an important area of mathematics, namely homotopy theory. The short citation for the Royal Swedish Academy of Sciences’ “2020 Rolf Schock Prize in logic and philosophy” says that it was awarded to Per Martin-Löf (shared with Dag Prawitz) “for the creation of constructive type theory.” In a longer statement, the prize committee recalls that constructive type theory is “a formal language in which it is possible to express constructive mathematics” (…) “[which] also functions as a powerful programming language and has had an enormous impact in logic, computer science and, recently, mathematics.”

A recent presentation by Michael Shulman (Homotopy type theory - A high-level language for invariant mathematics, March 2019) draws attention to the importance of Voevodsky’s homotopy type theory in finding a proper framework to deal with higher-order equalities in mathematics:

“Homotopy type theory is a high-level abstract framework for working with sameness.”

4From Martin-Löf’s first published article on type theory with identity types:

“The formal system that we shall erect consists of a certain number of mechanical rules for deriving symbolic expressions of the forms

\[ a \in A \]

and

\[ a \text{ conv } b \]

which are to be read a is a term with type symbol A and a converts into b, respectively. In the intended interpretation, a \(\in A\) will mean that a is an object of type A and a conv b that \(a =_{\text{def }} b\).” (13, p.87)

Church’s (1936) original formulation of the ‘conversion’ equality:

“We consider the three following operations on a well-formed formula:

I. To replace any part \(\lambda x[M]\) of a formula by \(\lambda y[S^x_y M]\) where \(y\) is a variable which does not occur in \(M\).

II. To replace any part of \(\{\lambda x[M]\}(N)\) of a formula by \(S^x_y M\), provided that the bound variables in \(M\) are distinct both from \(x\) and from the free variables in \(N\).

III. To replace any part of \(S^x_y M\) (not immediately following \(\lambda\)) of a formula by \(\{\lambda x[M]\}(N)\), provided that the bound variables in \(M\) are distinct both from \(x\) and from the free variables in \(N\).

Any finite sequence of these operations is called a conversion, and if \(B\) is obtainable from \(A\) by a conversion we say that \(A\) is convertible into \(B\), or “\(A\) conv \(B\).” (…)}
to suggest that these examples might bear witness to the impact of MLTT in mathematics by offering formal tools to calculate and prove fundamental groups, as well as allowing to make such calculations and proofs amenable to be dealt with by systems of formal mathematics and interactive theorem provers such as Coq, Lean, and similar ones.

2 Computational paths

In this section, we introduce the main working tool, an entity known as computational paths. In [32], we have seen that it is possible to interpret the identity type semantically, considering the terms as homotopical paths between two points of a space. Thus, inspired by the path-based approach of the homotopy interpretation, we can use a similar approach to define the identity type in type theory, this entity is known as computational path.

The interpretation will be similar to the homotopy case: a term \( p : \text{Id}_A(a,b) \) will be a computational path between terms \( a,b : A \), and such path will be the result of a sequence of rewrites. In what follows, we shall formally define the concept of a computational path. The main notion, i.e. proofs of equality statements as (reversible) sequences of rewrites, is not new, as it goes back to a paper entitled “Equality in labeled deductive systems and the functional interpretation of propositional equality”, presented in December 1993 at the 9th Amsterdam Colloquium, and published in the proceedings in 1994 [24].

Indeed, one of the most interesting aspects of the identity type is the fact that it can be used to construct higher structures. This is a rather natural consequence of the fact that it is possible to construct higher identities. For any \( a,b : A \), we have type \( \text{Id}_A(a,b) \). If this type is inhabited by any \( p,q : \text{Id}_A(a,b) \), then we have type \( \text{Id}_{\text{Id}_A(a,b)}(p,q) \). If the latter type is inhabited, we have a higher equality between \( p \) and \( q \) [6]. This concept is also present in computational paths. One can prove the equality between two computational paths \( s \) and \( t \) by constructing a third one between \( s \) and \( t \). We provide in this section a system of rules used to establish equalities between computational paths [21].

Another important question we seek to answer is one that arises naturally when talking about equality: Is there a canonical proof for a statement like \( t_1 = t_2 \)? In the language of computational paths, is there a normal path between \( t_1 \) and \( t_2 \) such that every other path can be reduced to it? In [36], it was proved that the answer is negative, this model also refutes the Uniqueness of Identity Proofs (UIP).

A function \( F \) of one positive integer is said to be \( \lambda \)-definable if it is possible to find a formula \( F \) such that, if \( F(m) = r \) and \( m \) and \( r \) are the formulas for which the integers \( m \) and \( r \) (written in Arabic notation) stand according to our abbreviations introduced above, then \( \{F\}(m) \cong r \).

Our approach differs from the one developed in the HoTT book [9] in that we do not need to simulate the path-space between those points via a certain coding process, since we have computational paths as part of the theory’s syntax, and they are an integral part of the whole formalisation framework in the tradition of the Curry-Howard interpretation. See, for example:

“To characterize a path space, the first step is to define a comparison fibration “code” that provides a more explicit description of the paths. There are several different methods for proving that such a comparison fibration is equivalent to the paths (we show a few different proofs of the same result in §8.1). The one we have used here is called the encode-decode method: the key idea is to define decode generically for all instances of the fibration (i.e. as a function \( \Pi_{(x:A+B)}\text{code}(x) \to (\text{inl}(a_0) = x) \)), so that path induction can be used to analyze \( \text{decode}(x, \text{encode}(x,p)) \).” (p.95)

There have been attempts at developing other methods to improve on the ‘encode-decode’ method, such as [11]:

“Often, we want to find out what specific equality types look like. This is directly the goal when calculating the homotopy groups of given types (as in the synthetic homotopy theory mentioned above), but it is also a necessary intermediate step for many other constructions. For a very concrete example, let us recall the calculation of the loop space of the circle \( S^1 \) by Licata and Shulman. This loop space of \( S^1 \), as defined above in (4), is by definition simply the equality type (base = base). Licata and Shulman introduce and explain the encode-decode method: in order go get started, they “guess” that the loop space in question is equivalent to the integers \( \mathbb{Z} \) (looking at the left side of (4), the intuition is that one can go around the loop clockwise any number of times, and negative numbers correspond to going counterclockwise).

(...)”

The encode-decode method has been employed successfully in a variety of cases. Going through the necessary steps can be somewhat tedious but it often at least partially mechanical. One main goal in this paper is to develop a different method to directly work with equality types of coequalizers and pushouts (and constructions based on them).

The proposal makes use of a path constructor called ‘glue’, which is built to simulate paths but is not part of the primitive constructions in HoTT, even if, as it is argued, it can be constructed out of the standard elimination operator \( J \) of the identity type.
2.1 Introducing computational paths

Before we go into details on computational paths, let us recall what motivated the introduction of computational paths to type theory. In type theory, our types are interpreted using the so-called Brower-Heyting-Kolmogorov Interpretation. That way, a semantic interpretation of formulas are not given by truth-values, but by the concept of proof as a primitive notion. Thus, we have [27]:

\[
\text{a proof of the proposition: is given by:}
\]
\[
A \wedge B \quad \text{a proof of } A \text{ and a proof of } B
\]
\[
A \lor B \quad \text{a proof of } A \text{ or a proof of } B
\]
\[
A \rightarrow B \quad \text{a function that turns a proof of } A \text{ into a proof of } B
\]
\[
\forall x^D. P(x) \quad \text{a function that turns an element } a \text{ into a proof of } P(a)
\]
\[
\exists x^D. P(x) \quad \text{an element } a \text{ (witness) and a proof of } P(a)
\]

Also, based on the Curry-Howard functional interpretation of logical connectives, one has [27]:

\[
\text{a proof of the proposition: has the canonical form of:}
\]
\[
A \wedge B \quad (p, q) \text{ where } p \text{ is a proof of } A \text{ and } q \text{ is a proof of } B
\]
\[
A \lor B \quad i(p) \text{ where } p \text{ is a proof of } A \text{ or } j(q) \text{ where } q \text{ is a proof of } B
\]
\[
A \rightarrow B \quad \lambda x.b(x) \text{ where } b(p) \text{ is a proof of } B
\]
\[
\forall x^A.B(x) \quad \Lambda x.f(x) \text{ where } f(a) \text{ is a proof of } B(a)
\]
\[
\exists x^A. B(x) \quad \varepsilon x.(f(x), a) \text{ where } a \text{ is a witness}
\]

(For a more detailed explanation of the last clause, see [25].)

Upon further inspection, there is one interpretation missing in the BHK-Interpretation. What constitutes a proof of \( t_1 = t_2 \)? In other words, what is a proof of an equality statement? In [27] it was proposed that an equality between these two terms should be a sequence of rewritings starting at \( t_1 \) and ending at \( t_2 \).

We answer this by proposing that an equality between those two terms should be a sequence of rewrites starting from \( t_1 \) and ending at \( t_2 \). Thus, we would have [27]:

\[
\text{a proof of the proposition: is given by:}
\]
\[
t_1 = t_2 \quad ?
\]
\[
\text{(Perhaps a sequence of rewrites starting from } t_1 \text{ and ending in } t_2?)
\]

We call computational path the sequence of rewrites between these terms.

2.2 Formal definition

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 [29]. 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 \( a \) by 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 rewrites) a term in another. For that reason, a computational path is also known as a sequence of rewrites. Nevertheless, before we formally define a computational path, we can take a look at one famous equality theory, the \( \lambda \beta \eta \) equality [7]:

**Definition 1** ([7]). *The \( \lambda \beta \eta \) -equality is composed by the following axioms:

\[
\alpha \quad \lambda x.M = \lambda y.M[y/x] \quad \text{if } y \notin FV(M);
\]
(β) \((λx.M)N = M[N/x]\):

(ρ) \(M = M\);

(η) \((λx.Mx) = M\ (x \notin FV(M))\).

And the following rules of inference:

\[
\frac{M = M'}{NM = NM'} \quad \frac{M = N}{N = P} \quad \frac{M = M'}{MN = M'N}
\]

\[
\frac{M = M'}{\lambda x.M = \lambda x.M'}
\]

**Definition 2** \([7]\). \(P\) is \(β\)-equal or \(β\)-convertible to \(Q\) (notation \(P =_β Q\)) 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, \(P =_β Q\) iff there exist \(P_0, \ldots, P_n (n \geq 0)\) such that \(P_0 \equiv P, P_n \equiv Q\), \((∀i \leq n−1)(P_i \triangleright_1 P_{i+1} ∨ P_{i+1} \triangleright_1 P_i ∨ P_i \equiv_a P_{i+1})\).

(Notice that equality has an existential force, which will be shown in the proof rules for the identity type\(^7\))

The same happens with \(λβη\)-equality:

\^{6}We adhere strictly to the approach of giving the definitional equalities as the basis for the construction of proof-objects of the identity type, something which may not be followed everywhere. For example, in \([43]\) (p.25) Voevodsky introduces \(β\)-equality as part of the ‘rules for the equivalence types’:

\[
[\text{smart0}] \Gamma \triangleright Q : T \text{type} \quad \Gamma, y : R \triangleright q : Q \quad \Gamma \triangleright h : eq_R(r, r') \quad \Gamma \triangleright eq(q, r/y), q(r'/y)
\]

and the same for \(ξ\)-equality:

\[
[\text{smart2}] \Gamma, y : R \triangleright e : eq_Q(q, q') \quad \Gamma \triangleright ex(e) : eq_{Q'}(q, q')
\]

\^{7}An anonymous referee has asked “What does it mean that equality has an existential force?” and the answer is that our rules are aimed at formalising the kind of reasoning embedded in the definition of equality between \(λ\)-terms from Church’s original definition of conversion, which says that two terms \(M\) and \(N\) are equal if there is a sequence of applications of the rewriting rules \((β, α, β^{-1})\) starting from \(M\) and arriving at \(N\). As for the elimination rules in natural deduction style for existential-content propositions, which uses a local assumption, it is worth noticing that in the elimination rules for Martin-Löf’s original \(Id\) type, the framing reflects the pattern of existential-like elimination:

\[
[\{P(t)\}] \quad \exists x P(x) \quad C
\]

even though the ‘entity’ which is at the center of the existential content does not appear explicitly:

\[
\begin{array}{llll}
\frac{[x : A]}{\exists x P(x) \quad C} & \quad a : A & b : A & c : Id_A(a, b) & d(x) : C(x, x, x(x)) \\
\frac{\frac{[x : A]}{Id\text{-elimination}}}{J(c, d) : C(a, b, c)}
\end{array}
\]

(provided \([x : A, y : A, z : Id_A(x, y)]\) leads to \(C(x, y, z)\) \(type\)). Notice the use of a new local assumption \([x : A]\) being introduced to arrive at a certain unspecified conclusion \(‘C(x, x, x(x))’\), with the usual provisions. The lack of an explicit entity witnessing the propositional equality renders the explanation of the ‘elimination’ operator ‘\(J\)’ a nontrivial challenge. A rather technical explanation is given by \([5]\) via weak factorization systems.

In an early draft entitled ‘Notes on homotopy \(λ\)-calculus’ \([43]\), Voevodsky also uses a rule for the \(eq_R\) equality type which ‘may be considered as an analog of the equality elimination rules in other dependent type systems’:

“Since equivalences can be “pushed through” all term expressions with the help of rule (15) this implies that we only need an analog of the identity elimination rule for the expressions \(Q = eq_R(x, y)\). This is achieved by our rule (17) which therefore may be considered as an analog of the equality elimination rules in other dependent type systems. From this point of view rule (18) corresponds to the \(β\)-conversion for the equality. We could have introduced a conversion instead of the equivalence \(ε(−)\) but this approach allows more flexibility in the models.”
Definition 3 (\(\lambda\beta\eta\)-equality [7]). The equality-relation determined by the theory \(\lambda\beta\eta\) is called \(=_{\beta\eta}\); that is, we define
\[ M =_{\beta\eta} N \iff \lambda\beta\eta \vdash M = N. \]

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

Example 2. The term \(M \equiv (\lambda x. (\lambda y. yx)(\lambda w. zw))v\) is \(\beta\eta\)-equal to \(N \equiv vz\) because of the sequence:
\[ (\lambda x. (\lambda y. yx)(\lambda w. zw))v, \ (\lambda x. (\lambda y. yx)z)v, \ (\lambda y. yv)z, \ vz \]
Now, turning 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:
\[ \beta((\lambda x. (\lambda y. yx)z)v, (\lambda y. yv)z) \]
The first one is equal to the fourth one based on the grounds:
\[ \beta((\lambda y. yv)z, vz) \]

The first one is equal to the fourth one based on the grounds:
\[ \tau(\eta((\lambda x. (\lambda y. yx)(\lambda w. zw))v, (\lambda x. (\lambda y. yx)z)v), \beta((\lambda x. (\lambda y. yx)z)v, (\lambda y. yv)z)) \]

The aforementioned theory establishes the equality between two \(\lambda\)-terms. Since we are working with computational objects as terms of a type, we can consider the following definition:

Definition 4. The equality theory of Martin Löf’s type theory has the following basic proof rules for the \(\Pi\)-type:

\[
\frac{N : A \quad M : B}{(\lambda x. M)N = M[N/x] : B[N/x]} \quad \frac{[x : A]}{\lambda x.M = \lambda x.M' : (\Pi x : A)B}
\]

\[
\frac{M : A}{M = M : A} \quad \frac{M = M' : A}{\lambda x. M = \lambda x. M' : (\Pi x : A)B}
\]

\[
\frac{M = N : A}{N = M : A} \quad \frac{N : A}{\lambda x. M = \lambda x. M' : (\Pi x : A)B}
\]

\[
\frac{N = M : A}{M = N : A} \quad \frac{N = P : A}{\lambda x. M = \lambda x. M' : (\Pi x : A)B}
\]

\[
\frac{M = (\Pi x : A)B}{M = P : A} \quad \frac{(\lambda x. M)N = M : (\Pi x : A)B}{x \notin \text{FV}(M)}
\]

We are finally able to formally define computational paths:

Definition 5. 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 it by \(a =_s b\).

As we have seen in Example 2, the composition of rewrites is an application of the rule \(\tau\). Since the change of bound variables is possible, each term is considered up to an \(\alpha\)-equivalence.

where the rules (17) and (18) are framed as:

\[
\frac{\Gamma \vdash R : \text{Type}}{\Gamma, x, y, z : R, \phi : e_{\text{eq}}(x, y), \psi : e_{\text{eq}}(x, z) \vdash s(\phi, \psi) : e_{\text{eq}}(\pi(id(x), \phi))} \quad (17)
\]

\[
\frac{\Gamma \vdash R : \text{Type}}{\Gamma, x, y : R, \phi : e_{\text{eq}}(x, y), \tau s(\phi) : e_{\text{eq}}(\pi(id(x), \phi), \phi)} \quad (18)
\]
2.3 Equality equations

One can use the aforementioned axioms to illustrate that computational paths establish the three fundamental equations of equality: the reflexivity, symmetry and transitivity:

\[
\begin{align*}
\frac{a =_{\text{t}} b : A \quad b =_{\text{u}} c : A}{a =_{\tau(t,u)} c : A} \quad & \text{transitivity} \\
\frac{a : A}{a =_{\rho} a : A} \quad & \text{reflexivity} \\
\frac{a =_{\text{t}} b : A}{b =_{\sigma(t)} a : A} \quad & \text{symmetry}
\end{align*}
\]

2.4 Identity type

We have said that one can formulate the identity type using computational paths. As we have seen, the best way to define any formal entity of type theory is by a set of natural deduction rules. Thus, we define our path-based approach as the following set of rules:

- Formation and Introduction rules \([27, 32]\):

\[
\frac{A \text{ type}}{Id_{A}(a,b) \text{ type}}
\]

\[
\frac{a =_{s} b : A}{s(a,b) : Id_{A}(a,b)}
\]

One can notice that our formation rule is exactly equal to the identity type in type theory. From terms \(a, b : A\), it is possible to claim that the identity type is inhabited only if there is a proof of equality between those terms, i.e., \(Id_{A}(a,b)\).

The difference starts with the introduction rule. In our approach, one can notice that we do not use a reflexive constructor \(r\). In other words, the reflexive path is not the main building block of our identity type. Instead, if we have a computational path \(a =_{s} b : A\), we introduce \(s(a,b)\) as a term of the identity type. That way, one should see \(s(a,b)\) as a sequence of rewrites and substitutions (i.e., a computational path) which would have started from \(a\) and arrived at \(b\).

- Elimination rule \([27, 32]\):

\[
\frac{m : Id_{A}(a,b) \quad h(g) : C}{\text{REW}(m, \dot{g}, h(g)) : C}
\]

---

8 An anonymous referee questioned ‘the distinction between “\(a =_{s} b : A\)” and “\(s : Id_{A}(a,b)\)”’, and the answer starts from Martin-Löf’s original distinction between propositions and judgements:

“If \(x\) and \(y\) are objects of one and the same type \(A\), then

\(I(x, y)\)

is a proposition, namely, the proposition that \(x\) and \(y\) are identical.” (p.81)

Further along the same line of reasoning:

“If \(a =_{\text{def}} b\), then \(I(a, b)\) is true, that is, \(a\) and \(b\) are identical” (p.86)

In our proposed formulation, the propositional equality is written as \(Id_{A}(x, y)\), and judgemental equality as \(a =_{s} b : A\) (where instead of simply saying that the latter comes from a definition equality, it carries an identifier ‘\(s\)’ denoting the rewriting path, i.e., the composition of (possibly several) definition equalities.

From ‘Truth of a proposition, evidence of a judgement, validity of a proof’ \([16]\):

“First of all, we have the notion of proposition. Second, we have the notion of truth of a proposition. Third, combining these two, we arrive at the notion of assertion or judgement.” (p.409)
Let us recall the notation being used. First, one should see $h(g)$ as a functional expression $h$ which depends on $g$. Also, one should notice the use of "" as an abstractor that binds the occurrences of the variable $g$ introduced in the local assumption $[a =_g b : A]$ as a kind of Skolem-type constant denoting the reason why $a$ was assumed to be equal to $b$.

We also introduce the constructor $\text{REWR}$. In a sense, it is similar to the constructor $J$ of the traditional approach, since both arise from the elimination rule of the identity type. The behavior of $\text{REWR}$ is simple. If from a computational path $g$ that establishes the equality between $a$ and $b$ one can construct $h(g) : C$, then if we also have this equality established by a term $C$, we can gather all this information in $\text{REWR}$ to construct $C$, eliminating the type $Id_A(a, b)$ in the process. The idea is that we can substitute $g$ for $m$ in $\dot{g}.h(g)$, resulting in $h(m/g) : C$. This behavior is established below by the reduction rule.

- Reduction rule

\[
\begin{align*}
\frac{a =_m b : A}{m(a, b) : Id_A(a, b)} & \quad \frac{Id - I}{h(g) : C} \quad \frac{Id - E}{\text{REWR}(m, \dot{g}.h(g)) : C} \\
\frac{[a =_g b : A]}{h(g) : C} \quad \frac{Id - E}{\text{REWR}(m, \dot{g}.h(g)) : C} \\
\frac{a =_m b : A}{h(m/g) : C}
\end{align*}
\]

- Induction rule:

\[
\begin{align*}
\frac{e : Id_A(a, b)}{t(a, b) : Id_A(a, b)} & \quad \frac{Id - I}{Id - E} \quad \frac{Id - E}{\text{REWR}(e, \dot{t}(a, b)) : Id_A(a, b)} \\
\frac{[a =_e b : A]}{t(a, b) : Id_A(a, b)} \quad \frac{Id - I}{Id - E} \quad \frac{Id - E}{\text{REWR}(e, \dot{t}(a, b)) : Id_A(a, b)} \\
\frac{e : Id_A(a, b)}{a =_e b : A}
\end{align*}
\]

Our introduction and elimination rules reassure the concept of equality as an existential force. In the introduction rule, we encapsulate the idea that a witness of an identity type $Id_A(a, b)$ only exists if there is a computational path establishing the equality of $a$ and $b$. Also, one can notice that this elimination rule is similar to the elimination rule of the existential quantifier.

### 2.5 Path-based examples

The objective of this subsection is to demonstrate how to put into practice the rules that we have just defined. The objective is to show the construction of terms of some important types. The constructions that we have chosen to build are the reflexive, transitive and symmetric type of the identity type. Those were not random choices. The main reason for having picked them is the fact that reflexive, transitive and symmetric types are essential to the process of building a groupoid model for the identity type. As we shall see, these constructions come naturally from simple computational paths constructed by the application of axioms of the equality of type theory.

Before we start the constructions, it is our opinion that it is essential to understand how to use the eliminations rules. The process of building a term of some type is a matter of finding the right reason. In the case of $J$, the reason is the correct $x, y : A$ and $z : Id_A(a, b)$ that generates the adequate $C(x, y, z)$. In our approach, the reason is the correct path $a =_g b$ that generates the adequate $g(a, b) : Id(a, b)$.

---

9The use of the variable-binding via a notation which differs from $\lambda$ has the purpose of avoiding imposing each and every property of $\lambda$-conversion for this particular abstraction. Perhaps in the same vein, Voevodsky introduces a $\theta$-abstraction in [44]:

"The notation $\theta y : \text{R}.(h, q)$ is chosen to emphasize that $y$ becomes a bound variable in this expression." (p.25)

Yet another notation for variable binding (abstraction) is used by Martin-Löf in the definition of the rule of $\Sigma$-elimination:

\[ (x \in A, y \in B(x)) (d(x, y) \in C((x, y))) \]
\[ \text{E}(c, (x, y) d(x, y)) \in C(c) \]

"Another notation for $\text{E}(c, (x, y) d(x, y))$ could be $(\text{E}x, y) (c, d(x, y))$, but we prefer the first since it shows more clearly that $x$ and $y$ become bound only in $d(x, y)$." (p.40)
2.5.1 Reflexivity

One could find strange the fact that we need to prove the reflexivity. Nevertheless, just remember that our approach is not based on the idea that reflexivity is the base of the identity type. As usual in type theory, a proof of something comes down to a construction of a term of a type. In this case, we need to construct a term of type $\Pi_{(a:A)}\text{Id}_A(a,a)$. The reason is extremely simple: from a term $a : A$, we obtain the computational path $a =_\rho a : A$ \([32]\):

$$
\frac{[a : A]}{a =_\rho a : A}
\frac{\rho(a,a) : \text{Id}_A(a,a)}{\lambda a.\rho(a,a) : \Pi_{(a:A)}\text{Id}_A(a,a)} \Pi - I
$$

2.5.2 Symmetry

The second proposed construction is the symmetry. Our objective is to obtain a term of type $\Pi_{(a:A)}\Pi_{(b:A)}(\text{Id}_A(a,b) \rightarrow \text{Id}_A(b,a))$.

We shall construct a proof using computational paths. As expected, we need to find a suitable reason. Starting from $a =_\iota b$, we could look at the axioms of Definition 4.1 to plan our next step. One of those axioms makes the symmetry clear: the $\sigma$ axiom. If we apply $\sigma$, we will obtain $b =_{\sigma(t)} a$. From this, we can then infer that $\text{Id}_A$ is inhabited by $(\sigma(t))(b,a)$. Now, it is just a matter of applying the elimination \([32]\):

$$
\frac{[a : A]}{[b : A]}
\frac{[a =_\iota b : A]}{\frac{[p(a,b) : \text{Id}_A(a,b)]}{\frac{[\sigma(t)](b,a) : \text{Id}_A(b,a)}{\text{Id} - I}}}
\frac{REWR R(p(a,b), t.(\sigma(t))(b,a)) : \text{Id}_A(a,b) \rightarrow \text{Id}_A(b,a) \rightarrow -I}{\text{Id} - E}
\frac{\lambda a.\lambda b.\lambda p.\text{REWR R}(p(a,b), t.(\sigma(t))(b,a)) : \Pi_{(a:A)}\Pi_{(b:A)}(\text{Id}_A(a,b) \rightarrow \text{Id}_A(b,a))}{\Pi - I}
\frac{\lambda a.\lambda b.\lambda p.\text{REWR R}(p(a,b), t.(\sigma(t))(b,a)) : \Pi_{(a:A)}\Pi_{(b:A)}(\text{Id}_A(a,b) \rightarrow \text{Id}_A(b,a))}{\Pi - I}
$$

2.5.3 Transitivity

The third and last construction will be the transitivity. Our objective is to obtain a term of type

$$
\Pi_{(a:A)}\Pi_{(b:A)}\Pi_{(c:A)}(\text{Id}_A(a,b) \rightarrow \text{Id}_A(b,c) \rightarrow \text{Id}_A(a,c)).
$$

To build our path-based construction, the first step, as expected, is to find the reason. Since we are trying to construct the transitivity, it is natural to think that we should start with paths $a =_\iota b$ and $b =_\iota c$ and then, from these paths, we should conclude that there is a path $z$ that establishes that $a =_\iota c$. To obtain $z$, we could try to apply the axioms of Definition 4.1. Looking at the axioms, one of them is exactly what we want: the axiom $\tau$. If we apply $\tau$ to $a =_\iota b$ and $b =_\iota c$, we will obtain a new path $\tau(t,u)$ such that $a =_{\tau(t,u)} c$. Using that construction as the reason, we obtain the following term \([32]\):
\[
\begin{array}{c}
\begin{array}{c}
\frac{[a \vdash b : A]}{[c : A]} \quad \frac{[b = a : A]}{[c = \tau(t, u) : A]}
\end{array}
\end{array}
\quad \begin{array}{c}
\frac{\tau(t, u)}{(a, c) : Id\ A(a, c)} \quad Id - E
\end{array}
\]

\[
\frac{[w(a, b) : Id\ A(a, b)]}{(s(b, c) : Id\ A(b, c))} \quad \begin{array}{c}
\frac{REWR(s(b, c), \hat{u}(\tau(t, u))(a, c)) : Id\ A(a, c)}{Id - E}
\end{array}
\]

\[
\frac{\lambda w. \lambda s. REWR(w(a, b), \hat{t} REWR(s(b, c), \hat{u}(\tau(t, u))(a, c))) : Id\ A(b, c) \rightarrow Id\ A(a, c)}{\Pi - I}
\]

\[
\frac{\lambda c. \lambda c. \lambda w. \lambda s. REWR(w(a, b), \hat{t} REWR(s(b, c), \hat{u}(\tau(t, u))(a, c))) : \Pi_{\beta, \Pi_{\alpha, A}}(Id\ A(a, b) \rightarrow Id\ A(b, c) \rightarrow Id\ A(a, c))}{\Pi - I}
\]

\[
\frac{\lambda a. \lambda b. \lambda c. \lambda w. \lambda s. REWR(w(a, b), \hat{t} REWR(s(b, c), \hat{u}(\tau(t, u))(a, c))) : \Pi_{\beta, \Pi_{\alpha, A}}(Id\ A(a, b) \rightarrow Id\ A(b, c) \rightarrow Id\ A(a, c))}{\Pi - I}
\]
As one can see, each step is simply composed of straightforward applications of introduction, elimination rules and abstractions. The only idea behind this construction is the simple fact that the axiom $\tau$ guarantees the transitivity of paths.

2.6 Term rewrite system

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 $a =_r b : A$ and a path $t$ that establishes that $a =_r b : A$. Consider that $s$ and $t$ are formed by distinct compositions of rewrites. Is it possible to conclude that there are cases where $s$ and $t$ should be considered equivalent? The answer is yes. Consider the following examples:

Example 3. Consider the path $a =_r b : A$. By the symmetric property, we obtain $b =_{\sigma(t)} a : A$. What if we apply the property again on the $\sigma(t)$ path? We would obtain a path $a =_{\sigma(\sigma(t))} b : A$. Since we applied the symmetry twice in succession, we have obtained a path that is equivalent to the initial path $t$. For that reason, we would like to conclude that the act of applying the symmetry twice in succession is a redundancy. We say that the path $\sigma(\sigma(t))$ reduces to path $t$.

Example 4. Consider the reflexive path $a =_r a : A$. If the symmetric axiom is applied, we end up with $a =_{\sigma(\sigma(t))} a : A$. Thus, the obtained path is equivalent to the initial one, since the symmetry was applied to the reflexive path. Therefore, $\sigma(\sigma(r))$ is a redundant way of expressing the path $\rho$. Thus, $\sigma(\sigma(r))$ should be reduced to $\rho$.

Example 5. Consider a path $a =_r b : A$. By applying the symmetry, one ends up with $b =_{\sigma(t)} a : A$. It is possible to take those two paths and apply the transitivity, ending up with $a =_{\tau(t, \sigma(t))} a$. Since the path $\tau$ is the inverse of the $\sigma(\tau)$, the composition of those two paths should be equivalent to the reflexive path. Thus, $\tau(t, \sigma(t))$ should be reduced to $\rho$.

As can be seen in the aforementioned examples, different paths should be considered equal if one is a redundant form of the other. The examples that we have just seen are straightforward and simple cases. Since the equality theory has a total of 7 axioms, the possibility of combinations which could generate redundancies is high. Fortunately, all possible redundancies were thoroughly mapped by [21]. In that work, a system that establishes all redundancies and creates rules which solve them was proposed. This system, known as $LND_{EQ} - TRS$, maps a total of 39 rules that solve redundancies.

2.7 $LND_{EQ} - TRS$

In this subsection, we provide the rules which compose the $LND_{EQ} - TRS$. All those rules originate from the mapping of redundancies between computational paths, as we have seen in the 3 previous examples.

2.7.1 Subterm substitution

Before we introduce the rewriting rules, it is important to introduce the concept of subterm substitution. In Equational Logic, the subterm substitution is given by the following inference rule [23]:

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

where $\theta$ is a substitution. One problem is that such rule does not respect the sub-formula property. To deal with that, [11] proposes two inference rules:

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

where $M, N$ and $O$ are terms.

As proposed in [27], we can define similar rules using computational paths, as follows:

$$\frac{x =_r C[y] : A \quad y =_s u : A'}{x =_{\text{sub}_{(r,s)}} C[u] : A} \quad \frac{x =_r u : A' \quad C[w] =_s u : A}{C[x] =_{\text{sub}_{(r,s)}} u : A}$$

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 $A$. 

12
In the rule above, \( C[u] \) should be understood as the result of replacing every occurrence of \( y \) by \( u \) in \( C \).  

2.7.2 Rewriting rules

In this subsection, our objective is to demonstrate all rewrite reductions and their associated rewriting rules. The aim is to analyze all possible occurrences of redundancies in proofs which involve the rules of rewritings.

We start with the transitivity:

**Definition 6** (reductions involving \( \tau \) [27]).

\[
\frac{x =_r y : A \quad y =_{\sigma(r)} x : A}{x =_{\tau(r,\sigma(r))} x : A} \quad \triangleright_{tr} \quad x =_\rho x : A
\]

\[
\frac{y =_{\sigma(r)} x : A \quad x =_r y : A}{y =_{\tau(\sigma(r),r)} y : A} \quad \triangleright_{tsr} \quad y =_\rho y : A
\]

\[
\frac{u =_r v : A \quad v =_\rho v : A}{u =_{\tau(r,\rho)} v : A} \quad \triangleright_{trr} \quad u =_r v : A
\]

\[
\frac{u =_\rho u : A \quad u =_r v : A}{u =_{\tau(\rho,r)} v : A} \quad \triangleright_{tlr} \quad u =_r v : A
\]

**Associated rewriting rules:**

\[
\tau(r,\sigma(r)) \triangleright_{tr} \rho
\]

\[
\tau(\sigma(r),r) \triangleright_{tsr} \rho
\]

\[
\tau(r,\rho) \triangleright_{trr} r
\]

\[
\tau(\rho,r) \triangleright_{tlr} r.
\]

These reductions can be generalized to transformations where the reasons \( r \) and \( \sigma(r) \) (transf. 1 and 2) and \( r \) and \( \rho \) (transf. 3 and 4) appear in some context, as illustrated by the following example: [27]:

**Example 6.**

\[
\frac{x =_r y : A \quad y =_{\sigma(r)} x : A}{i(x) =_{\xi_1(r)} i(y) : A + B \quad i(y) =_{\xi_1(\sigma(r))} i(x) : A + B}{i(x) =_{\tau(\xi_1(r),\xi_1(\sigma(r)))} i(x) : A + B} \quad \triangleright_{tr}
\]

\[
\frac{x =_r y : A \quad y =_{\sigma(r)} x : A}{i(x) =_{\xi_1(r)} i(y) : A + B}{i(x) =_{\xi_1(r)} i(y) : A + B}
\]

**Associated rewriting:** \( \tau(\xi_1(r),\xi_1(\sigma(r))) \triangleright_{tr} \xi_1(r). \)

---

10 In Martin-Löf’s original paper [13] (p.85) the rule for subterm substitution is framed as:

\[
a =_{\text{def}} c \quad \frac{b[a] =_{\text{def}} b[c]}{
\}

13
For the general context $C[]$:

**Definition 7.**

$$
\begin{align*}
[x : A] \\
\vdots \\
b(x) &=_{\tau} g(x) : B \\
a : A \quad \lambda x. b(x) &=_{\xi(r)} \lambda x. g(x) : A \to B \\
APP(\lambda x. b(x), a) &=_{\nu(\xi(r))} APP(\lambda x. g(x), a) : B \\
\\vdash_{\text{mxl}} a : A \\
b(a/x) &=_{\tau} g(a/x) : B
\end{align*}
$$

**Associated rewriting rule:**

$$
\nu(\xi(r)) \vdash_{\text{mxl}} r.
$$

**Definition 8** (reductions involving $\rho$ and $\sigma$ [27]).

$$
\begin{align*}
x &=_{\rho} x : A \\
x &=_{\sigma(\rho)} x : A \\
\vdash_{\text{sr}} x &=_{\rho} x : A
\end{align*}
$$

$$
\begin{align*}
x &=_{\tau} y : A \\
y &=_{\sigma(r)} x : A \\
x &=_{\sigma(\sigma(\tau))} y : A \\
\vdash_{\text{sr}} x &=_{\tau} y : A
\end{align*}
$$

**Associated rewritings:**

$$
\begin{align*}
\sigma(\rho) &\vdash_{\text{sr}} \rho \\
\sigma(\sigma(r)) &\vdash_{\text{sr}} r
\end{align*}
$$

**Definition 9** (Substitution rules [27]).

$$
\begin{align*}
u =_{\tau} C[x] : A \\
u =_{\text{sub}_c(\tau, \rho)} C[x] : A \\
\vdash_{\text{srl}} u =_{\tau} C[x] : A
\end{align*}
$$

$$
\begin{align*}
x &=_{\rho} x : A' \\
C[x] &=_{\text{sub}_c(\tau, \rho)} z : A \\
\vdash_{\text{srr}} C[x] =_{\rho} z : A
\end{align*}
$$

$$
\begin{align*}
z =_{s} C[y] : A \\
y =_{r} w : A' \\
y =_{\sigma(r)} y : D' \\
z =_{\text{sub}_c(s, r)} C[w] : D \\
z =_{\text{sub}_c(s, r, \sigma(r))} C[y] : A \\
\vdash_{\text{sls}} z =_{s} C[y] : A
\end{align*}
$$
\[
\begin{align*}
\frac{z = s \ C[y] : A \quad y = r \ w : A'}{z = s_{\mathsf{sub}(s,r)} C[w] : A} \quad \frac{w = s_{\sigma(r)} y : A'}{w = s_{\mathsf{sub}(s_{\sigma(r)},s_{\sigma(r)})} C[y] : A} \quad \triangleright_{\mathsf{sst}} \quad z = s \ C[y] : A
\end{align*}
\]

Associated rewritings:
\[
\begin{align*}
\mathsf{sub}_l(C[r], C[\rho]) & \triangleright_{\mathsf{slr}} C[r] \\
\mathsf{sub}_b(C[\rho], C[r]) & \triangleright_{\mathsf{sr}} C[r] \\
\mathsf{sub}_l(\mathsf{sub}_b(s, C[r]), C[\sigma(r)]) & \triangleright_{\mathsf{ss}} \mathsf{s} \\
\mathsf{sub}_l(\mathsf{sub}_b(s, C[\sigma(r)]), C[r]) & \triangleright_{\mathsf{ss}} \mathsf{s} \\
\mathsf{sub}_b(s, \mathsf{sub}_b(C[\sigma(s)], s_{\mathsf{sr}})) & \triangleright_{\mathsf{sr}} \mathsf{r} \\
\mathsf{sub}_b(C[\sigma(s)], \mathsf{sub}_b(C[s], r)) & \triangleright_{\mathsf{sr}} \mathsf{r}
\end{align*}
\]

Definition 10 ([27]).

\[
\begin{align*}
\beta_{\mathsf{recr}^* \times \mathsf{r}-\mathsf{red}} \quad & \quad x = r \ y : A \\
\langle x, z \rangle = s_{\xi_{s}(r)} (y, z) : A \times B \quad \triangleright_{\mathsf{intr}} \quad & \quad FST(\langle x, z \rangle) = \mu_1(\xi_{s}(r)) FST(\langle y, z \rangle) : A \quad \triangleright_{\mathsf{elim}} \triangleright_{\mathsf{mx2l}} \quad x = r \ y : A \\
\langle x, y \rangle = s_{\xi_{s}(r,s)} (x', z) : A \times B \quad \triangleright_{\mathsf{intr}} \quad & \quad FST(\langle x, y \rangle) = \mu_1(\xi_{s}(r,s)) FST(\langle x', z \rangle) : A \quad \triangleright_{\mathsf{elim}} \triangleright_{\mathsf{mx2l}} \quad x = r \ x' : A \\
\langle x, z \rangle = s_{\xi_{s}(r,s)} (y, w) : A \times B \quad \triangleright_{\mathsf{intr}} \quad & \quad \mathsf{SND}(\langle x, z \rangle) = \mu_2(\xi_{s}(r,s)) \mathsf{SND}(\langle y, w \rangle) : B \quad \triangleright_{\mathsf{elim}} \triangleright_{\mathsf{mx2r}} \quad z = s \ w : B \\
\langle x, z \rangle = s_{\xi_{s}(s)} (x, w) : A \times B \quad \triangleright_{\mathsf{intr}} \quad & \quad \mathsf{SND}(\langle x, z \rangle) = \mu_2(\xi_{s}(s)) \mathsf{SND}(\langle x, w \rangle) : B \quad \triangleright_{\mathsf{elim}} \triangleright_{\mathsf{mx2r}} \quad z = s \ w : B
\end{align*}
\]

Associated rewritings:
\[
\begin{align*}
\mu_1(\xi_{s}(r)) & \triangleright_{\mathsf{mx2l}} \mathsf{r} \\
\mu_1(\xi_{s}(r,s)) & \triangleright_{\mathsf{mx2l}} \mathsf{r}
\end{align*}
\]
\[\mu_2(\xi(r, s)) \vdash_{mx\, 2\, r} \; s\]
\[\mu_2(\xi_2(s)) \vdash_{mx\, 2\, r} \; s\]

\[\beta_{rewr}^- \text{ reduction}\]
\[
\begin{array}{l}
\frac{a =_r a' : A}{i(a) =_{\xi(r)} i(a') : A + B} \quad + \text{-intr} \quad [x : A] \\
\frac{f(x) =_s k(x) : C \quad g(y) =_u h(y) : C}{D(i(a), \tilde{x}f(x), \tilde{y}g(y)) =_{\mu(\xi(r), s, u)} D(i(a'), \tilde{x}k(x), \tilde{y}h(y)) : C} \quad + \text{-elim}
\end{array}
\]
\[\vdash_{\text{mx}3\text{l}} \quad a =_r a' : A \quad f(a/x) =_s k(a'/x) : C\]

\[\begin{array}{l}
\frac{b =_r b' : B}{j(b) =_{\xi_2(r)} j(b') : A + B} \quad + \text{-intr} \quad [x : A] \\
\frac{f(x) =_s k(x) : C \quad g(y) =_u h(y) : C}{D(j(b), \tilde{x}f(x), \tilde{y}g(y)) =_{\mu(\xi_2(r), s, u)} D(j(b'), \tilde{x}k(x), \tilde{y}h(y)) : C} \quad + \text{-elim}
\end{array}\]
\[\vdash_{\text{mx}3\text{r}} \quad b =_s b' : B \quad g(b/y) =_u h(b'/y) : C\]

**Associated rewritings:**
\[
\mu(\xi_1(r), s, u) \vdash_{\text{mx}3\text{l}} \; s \\
\mu(\xi_2(r), s) \vdash_{\text{mx}3\text{r}} \; u
\]

\[\beta_{rewr}^- \Pi \text{ reduction}\]
\[
\begin{array}{l}
\frac{[x : A]}{f(x) =_r g(x) : B(x)}
\end{array}
\]
\[
\begin{array}{l}
\frac{a : A \quad \lambda x. f(x) =_{\xi(r)} \lambda x. g(x) : \Pi x : A.B(x)}{APP(\lambda x. f(x), a) =_{\nu(\xi(r))} APP(\lambda x. g(x), a) : B(a)}
\end{array}
\]
\[\vdash_{\text{mx}1} \quad a : A \quad f(a/x) =_r g(a/x) : B(a)\]

**Associated rewriting:**
\[
\nu(\xi(r)) \vdash_{\text{mx}1} \; r
\]

\[\beta_{rewr^-} \Sigma \text{ reduction}\]
\[
\begin{array}{l}
\frac{a =_r a' : A \quad f(a) : B(a)}{\varepsilon x. (f(x), a) =_{\xi_1(r)} \varepsilon x. (f(x), a') : \Sigma x : A.B(x)}
\end{array}
\]
\[
\begin{array}{l}
\frac{[t : A, g(t) : B(t)]}{E(\varepsilon x. (f(x), a), \tilde{y}td(g,t)) =_{\mu(\xi_1(r), s)} E(\varepsilon x. (f(x), a'), \tilde{y}h(g,t)) : C}
\end{array}
\]
\[\vdash_{\text{mx}r} \quad a =_r a' : A \quad f(a) : B(a) \quad d(f/g, a/t) =_s h(f/g, a'/t) : C\]

\[
\begin{array}{l}
\frac{a : A \quad f(a) =_r i(a) : B(a)}{\varepsilon x. (f(x), a) =_{\xi_2(r)} \varepsilon x. (i(x), a) : \Sigma x : A.B(x)}
\end{array}
\]
\[
\begin{array}{l}
\frac{[t : A, g(t) : B(t)]}{E(\varepsilon x. (f(x), a), \tilde{y}td(g,t)) =_{\mu(\xi_2(r), s)} E(\varepsilon x. (i(x), a), \tilde{y}h(g,t)) : C}
\end{array}
\]
\[\vdash_{\text{mx}l} \quad a : A \quad f(a) =_r i(a) : B(a) \quad d(f/g, a/t) =_s h(i/g, a/t) : C\]

**Associated rewritings:**
\[
\mu(\xi_1(r), s) \vdash_{\text{mx}r} \; s \\
\mu(\xi_2(r), s) \vdash_{\text{mx}l} \; s
\]
Definition 11 ($\eta_{rewr}$ [27]).

\( \eta_{rewr} \times -reduction \)

\[
\begin{align*}
x =_r y : A \times B &\xrightarrow{\text{-elim}} x =_r y : A \times B \\
FST(x) =_{\mu_1(r)} FST(y) : A &\xrightarrow{\text{-elim}} \langle FST(x), SND(x) \rangle =_{\xi(\mu_1(r))} \langle FST(y), SND(y) \rangle : A \times B \\
\langle FST(x), SND(x) \rangle &\xrightarrow{\text{\&}_\text{max}} \langle FST(x), SND(x) \rangle =_{\xi(\mu_1(r), \mu_2(r))} \langle FST(y), SND(y) \rangle : A \times B
\end{align*}
\]

\( \eta_{rewr} + -reduction \)

\[
\begin{align*}
c = t d : A + B &\xrightarrow{\text{-intr}} [a_1 =_r a_2 : A] + [b_1 =_s b_2 : B] \\
\mu(t, \xi_1(r), \xi_2(s)) =_{\xi(\mu_1(r))} D(c, a_1i(a_1), b_1j(b_1)) &\xrightarrow{\text{-intr}} D(c, a_1i(a_1), b_1j(b_1)) =_{\mu(t, \xi_1(r), \xi_2(s))} D(d, a_2i(a_2), b_2j(b_2)) \\
\mu(t, \xi_1(r), \xi_2(s)) &\xrightarrow{\text{\&}_\text{max}} \mu(t, \xi_1(r), \xi_2(s)) =_{\xi(\mu_1(r))} \mu(t, \xi_1(r), \xi_2(s))
\end{align*}
\]

where \( c \) and \( d \) do not depend on \( x \).

\( \Pi - \eta_{rewr} -reduction \)

\[
\begin{align*}
[t : A] &\xrightarrow{\text{\&}_\text{max}} \mu(t, \xi_1(r), \xi_2(s)) =_{\xi(\mu_1(r))} \mu(t, \xi_1(r), \xi_2(s))
\end{align*}
\]

\( \Sigma - \eta_{rewr} -reduction \)

\[
\begin{align*}
\mu(t, \xi_1(r), \xi_2(s)) &\xrightarrow{\text{\&}_\text{max}} \mu(t, \xi_1(r), \xi_2(s)) =_{\xi(\mu_1(r))} \mu(t, \xi_1(r), \xi_2(s))
\end{align*}
\]

Associated rewritings:
\( \xi(\mu_1(r), \mu_2(r)) \xrightarrow{\text{\&}_\text{max}} r \)
\( \mu(t, \xi_1(r), \xi_2(s)) \xrightarrow{\text{\&}_\text{max}} t \)
\( \xi(\nu(r)) \xrightarrow{\text{\&}_\text{xmr}} r \)
\( \mu(s, \xi_2(r)) \xrightarrow{\text{\&}_\text{mxtr}} s \)

Definition 12 ($\sigma$ and $\tau$ [27]).

\[
\begin{align*}
x =_r y : A &\xrightarrow{\text{\&}_\text{stss}} y =_s w : A \\
x =_{\tau(r, s)} w : A &\xrightarrow{\text{\&}_\text{stss}} x =_{\tau(r, s)} w : A \\
\end{align*}
\]

Associated rewriting:
\( \sigma(\tau(r, s)) \xrightarrow{\text{\&}_\text{stss}} \tau(\sigma(s), \sigma(r)) \)
Definition 13 (σ and sub [27]).

\[
\begin{align*}
\frac{x = r, C[y] : A}{x = \text{sub}_b(r, s) C[y] : A} & \triangleright \text{ssbr} \\
\frac{w = \sigma(b, s) w : A}{w = \text{sub}_b(\sigma(s), \sigma(r)) w : A} & \triangleright \text{ssbl}
\end{align*}
\]

\[
\begin{align*}
\frac{x = r, y : A'}{\frac{C[x] = \text{sub}_b(r, s) w : A}{w = \sigma(b, s) C[x] : D}} & \triangleright \text{ssbr} \\
\frac{\frac{y = s w : A'}{y = \sigma(r) x : A}}{\frac{C[y] = \text{sub}_b(\sigma(s), \sigma(r)) C[y] : A}{x = r, C[y] : A}} & \triangleright \text{ssbl}
\end{align*}
\]

Associated rewritings:
\[
\begin{align*}
\sigma(\text{sub}_b(r, s)) & \triangleright \text{ssbr} \quad \sigma(\sigma(s), \sigma(r)) \\
\sigma(\text{sub}_b(r, s)) & \triangleright \text{ssbl} \quad \sigma(\sigma(s), \sigma(r))
\end{align*}
\]

Definition 14 (σ and ξ [27]).

\[
\begin{align*}
\frac{x = r, y : A}{i(x) = \xi(r) f(y) : A + B} & \triangleright \text{sx} \\
\frac{y = \sigma(r) x : A}{i(y) = \xi(\sigma(r)) i(x) : A + B}
\end{align*}
\]

\[
\begin{align*}
\frac{x = r, y : A}{\frac{\langle y, w \rangle = \sigma(\xi(r)) \langle x, z \rangle : A \times B}{[x : A]}} & \triangleright \text{sxxs} \\
\frac{y = \sigma(r) z : B}{\frac{\langle y, w \rangle = \xi(\sigma(r)) \langle x, z \rangle : A \times B}{[x : A]}}
\end{align*}
\]

\[
\begin{align*}
\frac{\lambda x. f(x) = \sigma(\xi(s)) \lambda x. f(x) : \Pi x : A.B(x)}{\lambda x. f(x) = g(x) : B(x)} & \triangleright \text{ssms} \\
\frac{\lambda x. f(x) = \sigma(\xi(s)) \lambda x. f(x) : \Pi x : A.B(x)}{\lambda x. g(x) = \sigma(\xi(s)) \lambda x. f(x) : B(x)}
\end{align*}
\]

Associated rewritings:
\[
\begin{align*}
\sigma(\xi(r)) & \triangleright \text{sx} \quad \xi(\sigma(r)) \\
\sigma(\xi(r), s) & \triangleright \text{sxxs} \quad \xi(\sigma(r), \sigma(s)) \\
\sigma(\xi(s)) & \triangleright \text{ssms} \quad \xi(\sigma(s))
\end{align*}
\]

Definition 15 (σ and µ [27]).

\[
\begin{align*}
\frac{x = r, y : A \times B}{\text{FST}(x) = \mu(x) \text{FST}(y) : A} & \triangleright \text{sm} \\
\frac{\text{FST}(y) = \sigma(\mu(x)) \text{FST}(x) : A}{\text{FST}(y) = \sigma(\mu(x)) \text{FST}(x) : A}
\end{align*}
\]

\[
\begin{align*}
\frac{x = r, y : A \times B}{\text{SND}(x) = \mu(y) \text{SND}(y) : A} & \triangleright \text{sm} \\
\frac{\text{SND}(y) = \sigma(\mu(y)) \text{SND}(x) : A}{\text{SND}(y) = \sigma(\mu(y)) \text{SND}(x) : A}
\end{align*}
\]
\[ x =_r y : A \quad f =_r g : A \to B \]
\[ \text{APP}(f, x) =_{\mu(s, r)} \text{APP}(g, y) : B \]
\[ \text{APP}(g, y) =_{\sigma(\mu(s, r))} \text{APP}(f, x) : B \]

\[ \begin{array}{ll}
[ s : A ] & [ t : B ] \\
\end{array} \]
\[ x =_r y : A + B \quad d(s) =_u f(s) : C \quad e(t) =_v g(t) : C \\
D(x, \delta d(s), \epsilon e(t)) =_{\mu(r, u, v)} D(y, \delta f(s), \epsilon f(t)) : C \\
D(y, \delta f(s), \epsilon f(t)) : C =_{\sigma(\mu(r, u, v))} D(x, \delta d(s), \epsilon e(t)) : C \\
\]

\[ \begin{array}{ll}
[ s : A ] & [ t : B ] \\
\end{array} \]
\[ x =_r y : A + B \quad d(s) =_u f(s) : C \quad e(t) =_v g(t) : C \\
D(x, \delta d(s), \epsilon e(t)) =_{\mu(r, u, v)} D(y, \delta f(s), \epsilon f(t)) : C \\
D(y, \delta f(s), \epsilon f(t)) =_{\sigma(\mu(r, u, v))} D(x, \delta d(s), \epsilon e(t)) : C \\
\]

Associated rewritings:
\[ \sigma(\mu_1(r)) \triangleright_{sm} \mu_1(\sigma(r)) \]
\[ \sigma(\mu_2(r)) \triangleright_{sm} \mu_2(\sigma(r)) \]
\[ \sigma(\mu(s, r)) \triangleright_{smss} \mu(\sigma(s), \sigma(r)) \]
\[ \sigma(\mu(r, u, v)) \triangleright_{smss} \mu(\sigma(r), \sigma(u), \sigma(v)) \]

**Definition 16** (\( \tau \) and \( \text{sub} \) [27]).

\[ x =_r C[y] : A \quad y =_s w : A' \]
\[ C[w] =_t z : A \]
\[ x =_{\tau(\text{sub}_r(s), t)} z : A \]

\[ \begin{array}{ll}
[ s : A ] & [ t : B ] \\
\end{array} \]
\[ x =_r y : A + B \quad d(s) =_u f(s) : C \quad e(t) =_v g(t) : C \\
D(y, \delta f(s), \epsilon f(t)) =_{\mu(r, u, v)} D(x, \delta d(s), \epsilon e(t)) : C \\
D(x, \delta d(s), \epsilon e(t)) =_{\sigma(\mu(r, u, v))} D(y, \delta f(s), \epsilon f(t)) : C \\
\]
\[ x =_\tau C[z] : A \quad z =_s w : A' \]
\[ x =_\tau (r, sub_b(r, s)) C[w] : A \]
\[ x =_\tau (r, sub_b(r, s)) C[z] : A \]

\( \Rightarrow_{tshl} \)
\[ x =_\tau C[z] : A \quad z =_s w : A' \]
\[ x =_{sub_r(r, s)} C[w] : A \]

\[ x =_\tau C[w] : A \quad w =_s z : A' \]
\[ x =_{sub_r(r, s)} C[z] : A \]

\( \Rightarrow_{tshlr} \)
\[ x =_\tau C[w] : D \quad w =_s z : A' \]
\[ x =_{sub_r(r, s)} C[z] : A \]

\[ x =_\tau y : A \quad y =_\tau w : A \quad w =_s z : A \]
\[ x =_\tau (t, r, s) z : A \]
\[ \Rightarrow_{tt} \]
\[ x =_\tau y : A \quad y =_\tau w : A \quad w =_s z : A \]
\[ x =_\tau (t, r, s) z : A \]

**Definition 17** (\( \tau \) and \( \tau \) \[27\]).

**Associated rewritings:**

\[ \tau(sub_b(r, s), t) \Rightarrow_{tshl} \tau(r, sub_b(s, t)) \]
\[ \tau(sub_b(s, t), u) \Rightarrow_{tshlt} sub_b(s, \tau(t, u)) \]
\[ \tau(r, sub_b(r, s)) \Rightarrow_{tshlr} sub_b(r, s) \]
\[ \tau(r, sub_b(s, r)) \Rightarrow_{tshlt} sub_b(r, s) \]
\[ \tau(\tau(t, r), s) \Rightarrow_{tt} \tau(t, r, s) \]

Thus, we put together all those rules to compose our rewrite system:

**Definition 18** (\( LNEQ - TRS \[27\]).

1. \( \sigma(\rho) \Rightarrow_{tr} \rho \)
2. \( \sigma(\sigma(r)) \Rightarrow_{ss} r \)
3. \( \tau(C[r], C[\sigma(r)]) \Rightarrow_{tr} C[\rho] \)
4. \( \tau(\sigma(C[r]), C[r]) \Rightarrow_{tst} C[\rho] \)
5. \( \tau(C[r], C[\rho]) \Rightarrow_{ttr} C[r] \)
6. \( \tau(\sigma(C[r]), C[r]) \Rightarrow_{tr} C[\rho] \)
7. \( sub_b(C[r], C[\rho]) \Rightarrow_{s} C[r] \)
8. \( sub_b(C[\rho], C[r]) \Rightarrow_{s} C[\rho] \)
9. \( sub_b(sub_b(s, C[r]), C[\sigma(r)]) \Rightarrow_{sls} s \)
10. \( sub_b(sub_b(s, C[\sigma(r)]), C[r]) \Rightarrow_{sls} s \)
11. \( sub_b(C[s], sub_b(C[\sigma(s)], r)) \Rightarrow_{srs} r \)
12. \( sub_b(C[\sigma(s)], sub_b(C[s], r)) \Rightarrow_{srs} r \)
13. \( \mu_1(\xi_1(r)) \Rightarrow_{mx2l1} r \)
14. \( \mu_1(\xi_1(s, s)) \Rightarrow_{mx2l2} r \)
15. \( \mu_2(\xi_1(r, s)) \Rightarrow_{mx2r1} s \)
16. \( \mu_2(\xi_2(s)) \Rightarrow_{mx2r1} s \)
17. \( \mu_1(\xi_1(r), s, u) \Rightarrow_{mx3l} s \)
18. \( \mu_1(\xi_2(r), s, u) \Rightarrow_{mx3r} u \)
19. \( \nu(\xi(r)) \Rightarrow_{mxl} r \)
20. \( \mu_1(\xi_1(r), s) \Rightarrow_{mxr} s \)
21. \( \xi(\mu_1(r), \mu_2(r)) \Rightarrow_{mx} r \)
We are interested in the following theorem \[31, 27\]:

When we talk about these kinds of systems, two questions emerge: Does every computational path have a normal form? And if a computational path has a normal form, is it unique? To demonstrate that it has a normal form, one must prove

**Theorem 1**

The proofs uses a special kind of ordering, known as *recursive path ordering*, proposed by \[4\]:

**Definition 19** (Recursive path ordering \[4, 27\]). Let \(>\) be a partial ordering on a set of operators \(F\). The recursive path ordering \(>^*\) on the set \(T(F)\) of terms over \(F\) is defined recursively as follows:

\[
s = f(s_1, \ldots, s_m) >^* g(t_1, \ldots, t_n) = t,
\]

if and only if

1. \(f = g\) and \(\{s_1, \ldots, s_m\} \supseteq^* \{t_1, \ldots, t_n\}\), or
2. \(f > g\) and \(\{s\} \supseteq^* \{t_1, \ldots, t_n\}\), or
3. \(f \nsubseteq g\) and \(\{s_1, \ldots, s_m\} \supseteq^* \{t\}\)

where \(\supseteq^*\) is the extension of \(>^*\) to multisets.

This definition uses the notion of partial ordering in multisets. A given partial ordering \(>\) on a set \(S\) may be extended to a partial ordering \(\supseteq\) on finite multisets of elements of \(S\), wherein a multiset is reduced by removing one or more elements and replacing them with any finite number of elements, each one smaller than one of the elements removed \[4\].

---

2.8 Normalization

In the previous subsection, we have seen a system of rewrite rules that resolves reductions in a computational path. When we talk about these kinds of systems, two questions emerge: Does every computational path have a normal form? And if a computational path has a normal form, is it unique? To demonstrate that it has a normal form, one must prove that every computational path terminates, i.e., that after a finite number of rewrites, one will end up with a path that does not have any additional reduction. To prove that it is unique, one must illustrate that the system is confluent. In other words, if one has a path with 2 or more reductions, they must demonstrate that the choice of the rewrite rule does not matter. In the end, one will always obtain the same end-path without any redundancies.

2.8.1 Termination

We are interested in the following theorem \[31, 27\]:

**Theorem 1** (Termination property for LND\(_{EQ}\) – TRS). LND\(_{EQ}\) – TRS is terminating.

The proofs uses a special kind of ordering, known as *recursive path ordering*, proposed by \[4\]:

**Definition 19** (Recursive path ordering \[4, 27\]). Let \(>\) be a partial ordering on a set of operators \(F\). The recursive path ordering \(>^*\) on the set \(T(F)\) of terms over \(F\) is defined recursively as follows:

\[
s = f(s_1, \ldots, s_m) >^* g(t_1, \ldots, t_n) = t,
\]

if and only if

1. \(f = g\) and \(\{s_1, \ldots, s_m\} \supseteq^* \{t_1, \ldots, t_n\}\), or
2. \(f > g\) and \(\{s\} \supseteq^* \{t_1, \ldots, t_n\}\), or
3. \(f \nsubseteq g\) and \(\{s_1, \ldots, s_m\} \supseteq^* \{t\}\)

where \(\supseteq^*\) is the extension of \(>^*\) to multisets.
Thus, one can prove the termination property by demonstrating that in all rules $e \rightarrow d$ of the system, one has that $e >^* d$. We also need to define the precedence ordering on the rewrite operators. We define it as follows [27, 31]:

\[
\begin{align*}
\sigma &> \tau > \rho, \\
\sigma &> \xi, \\
\sigma &> \xi_\wedge, \\
\sigma &> \xi_1, \\
\sigma &> \xi_2, \\
\sigma &> \mu, \\
\sigma &> \mu_1, \\
\sigma &> \mu_2, \\
\sigma &> \text{sub}_L, \\
\sigma &> \text{sub}_R, \\
\tau &> \text{sub}_L 
\end{align*}
\]

Thus, one can prove the termination by evidencing that for every rule of $e \rightarrow d$ of LND$_{EQ} - TRS$, $e >^* d$. For almost every rule this is a straightforward and tedious process. We are not going to display all those steps in this work, but we can give the proof of two examples.

26. $\sigma(\text{sub}_L(r, s)) >^* \text{sub}_R(\sigma(s), \sigma(r))$ :

- $\sigma > \text{sub}_R$ from the precedence ordering on the rewrite operators.
- $\{\sigma(\text{sub}_L(r, s))\} >^* \{\sigma(r), \sigma(r)\}$ :
  - $\sigma(\text{sub}_L(r, s)) >^* \sigma(s)$ and $\sigma(\text{sub}_L(r, s)) >^* \sigma(r)$:
    \[
    \begin{align*}
    \sigma &= \sigma \\
    \{\text{sub}(r, s)\} &\gg \{s\} \text{ from the subterm condition.} \\
    \{\text{sub}(r, s)\} &\gg \{r\} \text{ from the subterm condition.}
    \end{align*}
    \]

27. $\sigma(\text{sub}_R(r, s)) \triangleright \text{sub}_L(\sigma(s), \sigma(r))$ :

- $\sigma > \text{sub}_L$ from the precedence ordering on the rewrite operators.
- $\{\sigma(\text{sub}_R(r, s))\} >^* \{\sigma(r), \sigma(r)\}$ :
  - $\sigma = \sigma$
  - $\{\text{sub}_R(r, s)\} \gg \{s\}$ from the subterm condition.
  - $\{\text{sub}_R(r, s)\} \gg \{r\}$ from the subterm condition.

All other proofs can be verified at [31].

2.8.2 Confluence

Before we go to the proof of confluence, one needs to observe that LND$_{EQ} - TRS$ is a conditional term rewriting system. This means that some rules can only be applied if the terms of the associated equation follow some rules. For example, for the rule $\mu_1 (\xi_\wedge (r, s)) \triangleright \text{sub}_R r$, it is necessary to have a $\beta$-Reduction such as $FST \langle x, y \rangle$. With that in mind, we have the following definition [31]:

Definition 20 (Conditional term rewriting system). In conditional term rewriting systems, the rules have conditions attached, which must be true for the rewrite to occur. For example, a rewrite rule $e \rightarrow d$ with condition $C$ is expressed as:

\[ C | e \rightarrow d \]

To prove the confluence, it is necessary to analyze all possible critical pairs using the superposition algorithm proposed by [10]. Thus, there should not be any divergent critical pair. For example, we can take the superposition of rules 1 and 2, obtaining: $\sigma(\sigma(\rho))$. We have two possible rewrites [31]:

- $\sigma(\sigma(\rho)) \triangleright_{sr} \sigma(\rho) \triangleright_{sr} \rho$
- $\sigma(\sigma(\rho)) \triangleright_{ss} \rho$

As can be seen, we ended up with the same term $\rho$. Thus, no divergence has been generated.

One should compare every pair of rules to find all critical pairs and see if there are any divergences. If some divergence occurs, the superposition algorithm proposed by [10] illustrates how to add new rules to the system in such a way
that it becomes confluent. As a matter of fact, that was the reason why rules 38 and 39 of \( LND_{EQ - TRS} \) have been introduced to the system \([27]\):

38. \( \tau(C[u], \tau(C[\sigma(u)], v)) \triangledown_{ttts} v \)
39. \( \tau(C[\sigma(u)], \tau(C[u], v)) \triangledown_{ttst} u \).

Those two rules introduced the following reductions to the system \([31]\):

\[
\begin{align*}
  x &= s \ u : D \\
  u &= \tau(u) x : D \\
  x &= v \ w : D
  \end{align*}
\]

\[
\triangledown_{ttts} x \triangledown_{vt} w
\]

\[
\begin{align*}
  x &= s \ w : D \\
  w &= \tau(w) x : D \\
  x &= s \ z : D \\
  w &= \tau(w) z : D
  \end{align*}
\]

\[
\triangledown_{ss} w \triangledown_{vt} z
\]

A full proof of confluence can be found in \([21,28,23,31]\).

### 2.8.3 Normalization procedure

We can now provide two normalization theorems:

**Theorem 2** (normalization \([31]\)). Every derivation in the \( LND_{EQ - TRS} \) converts to a normal form.

**Proof.** Direct consequence of the termination property.

**Theorem 3** (strong normalization \([31]\)). Every derivation in the \( LND_{EQ - TRS} \) converts to a unique normal form.

**Proof.** Direct consequence of the termination and confluence properties.

In this sense, every proof can be reduced to a normal one. To do so, one should identify the redundancies and, based on the rewrite rules, a proof can be constructed without any redundancies. We demonstrate this in the following example \([31]\):

\[
\begin{align*}
  f(x, z) &= s \ f(w, y) : D \\
  f(w, y) &= \tau(f(x, z)) : D \\
  f(w, y) &= \subL(f(x, z), \tau(f(x, z))) : D \\
  f(c, z) &= \tau(f(w, y)) : D \\
  f(c, z) &= \subL(f(c, z), \tau(f(w, y))) : D \\
  y &= t \ b : D
  \end{align*}
\]

This deduction generates the following path: \( \subL(\tau(f(w, y)), t) \). This path is not in normal form, having two redundancies \([31]\):

\[
\begin{align*}
  \subL(\tau(f(w, y)), t) &\triangledown_{ss} \subL(\tau(f(w, y)), t)
  \end{align*}
\]
Thus, we can identify those reductions and conceive a deduction without any redundancies:  

\[
\begin{align*}
  x & =_r c : D \\
  c & =_{\sigma(r)} x : D \\
  f(x, z) & =_s f(w, y) : D \\
  f(c, z) & =_{\text{sub}_{\text{R}}(\sigma(r), s)} f(w, y) : D \\
  y & =_t b : D \\
  f(c, z) & =_{\text{sub}_{\text{L}}(\text{sub}_{\text{R}}(\sigma(r), s), t)} f(w, b) : D 
\end{align*}
\]

2.9 Rewrite equality

As we have just seen, the LND\(_{\text{EQ}} -\)TRS has 39 rewrite rules. We call each rule a rewrite rule (abbreviation: rw-rule). We provide the following definition:

**Definition 21 (Rewrite Rule).** An rw-rule is any of the rules defined in LND\(_{\text{EQ}} -\)TRS.

Similarly to the \(\beta\)-reduction of \(\lambda\)-calculus, we have a definition for rewrite reduction:

**Definition 22 (Rewrite reduction).** Let \(s\) and \(t\) be computational paths. We say that \(s \triangleright_{\text{rw}} t\) (read as: \(s\) rw-contracts to \(t\)) if we can obtain \(t\) from \(s\) by the application of only one rw-rule. If \(s\) can be reduced to \(t\) by finite number of rw-contractions, then we say that \(s \equiv_{\text{rw}} t\) (read as \(s\) rw-reduces to \(t\)).

We shall also define rewrite contractions and equality:

**Definition 23 (Rewrite contraction and equality).** Let \(s\) and \(t\) be computational paths. We say that \(s =_{\text{rw}} t\) (read as: \(s\) is rw-equal to \(t\)) if \(t\) can be obtained from \(s\) by a finite (perhaps empty) series of rw-contractions and reversed rw-contractions. In other words, \(s =_{\text{rw}} t\) if there exists a sequence \(R_0, \ldots, R_n\), with \(n \geq 0\), such that

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

A fundamental result is the fact that rewrite equality is an equivalence relation:

**Proposition 1.** Rewrite equality is transitive, symmetric and reflexive. \(\square\)

Rewrite reduction and equality play fundamental roles in the groupoid model of a type based on computational paths, as we shall see hereafter.

2.10 LND\(_{\text{EQ}}\) -TRS(2)

Until now, we have concluded in this subsection that there exist redundancies which are resolved by a system called LND\(_{\text{EQ}} -\)TRS. This system establishes rules that reduce these redundancies. Moreover, we have concluded that these redundancies are simply redundant uses of the equality axioms shown in section 2. In fact, since these axioms only define an equality theory for type theory, we can be more specific and say that these are redundancies of the equality of type theory. As we have mentioned, the LND\(_{\text{EQ}} -\)TRS has a total of 39 rules [21][27].

Since the rw-equality is based on the rules of LND\(_{\text{EQ}} -\)TRS, one can imagine the high number of redundancies that rw-equality could cause. In fact, a thorough study of all the redundancies caused by these rules led to the work done in [36], which is solely interested in the redundancies caused by the fact that rw-equality is transitive, reflexive and symmetric with the addition of only one specific rw2-rule. Thus, a system called LND\(_{\text{EQ}} -\)TRS\(_2\) was created, which resolves all the redundancies caused by rw-equality (in the same way that LND\(_{\text{EQ}} -\)TRS resolves all the redundancies caused by equality).

Since we know that rw-equality is transitive, symmetric and reflexive, it should have the same redundancies that the equality had involving only those properties. Given that rw-equality is merely a sequence of rw-rules (which is also similar to equality, since equality is only a computational path, i.e., a sequence of identifiers), we could identify these sequences. Thus, if \(s\) and \(t\) are rw-equal because there exists a sequence \(\theta : R_0, \ldots, R_n\) that justifies the rw-equality, then we can write that \(s =_{\text{rw}} t\). Thus, by using rw-equality, we are able to rewrite all the rules which originated the ones involving \(\tau, \sigma\) and \(\rho\). For example, we have [32]:

- (Rewrite contraction and equality [32])
- (Rewrite Rule [32])
- (Definition 21)
- (Definition 22)
- (Definition 23)
- (Proposition 1)

PREPRINT
Therefore, we obtain the rule \( tt_2 \) which resolves one of the redundancies caused by the transitivity of \( rw \)-equality (the 2 in \( tt_2 \) indicates that it is a rule that resolves a redundancy of \( rw \)-equality). In fact, using the same reasoning, we can obtain, for \( rw \)-equality, all the redundancies that we have shown in Definition \([32]\). In other words, we have \( tr_2, trr_2, tlr_2, slr_2, ss_2 \) and \( tt_2 \). Since we now are provided with rules of \( LND_{EQ} - TRS_2 \), we can use all the concepts that we have just defined for \( LND_{EQ} - TRS \). The only difference is that instead of having \( rw \)-rules and \( rw \)-equality, we have \( rw_2 \)-rules and \( rw_2 \)-equality.

There is an important rule specific to this system. It stems from the fact that transitivity of reducible paths can be reduced in different ways, but generating the same result. For example, consider the simple case of \( \tau(s, t) \) and consider that it is possible to reduce \( s \) to \( s' \) and \( t \) to \( t' \). There are two possible \( rw \)-sequences which reduce this case: The first one is \( \theta : \tau(s, t) \triangleright_{1rw} \tau(s', t) \triangleright_{1rw} \tau(s', t') \) and the second \( \theta' : \tau(s, t) \triangleright_{1rw} \tau(s, t') \triangleright_{1rw} \tau(s', t') \). Both \( rw \)-sequences obtained the same result in similar ways, the only difference being the choices that have been made at each step. Since the variables, when considered individually, followed the same reductions, these \( rw \)-sequences should be considered redundant relative to each other and, for that reason, there should be an \( rw_2 \)-rule that establishes this reduction. This rule is called independence of choice and is denoted by \( cd_2 \). Since we already understand the necessity of such a rule, we can define it formally:

**Definition 24** (Independence of choice \([32]\)). Let \( \theta \) and \( \phi \) be \( rw \)-equalities expressed by two \( rw \)-sequences: \( \theta : \theta_1,...,\theta_m \) with \( n \geq 1 \) and \( \phi : \phi_1,...,\phi_m \), with \( m \geq 1 \). Let \( T \) be the set of all possible \( rw \)-equalities from \( \tau(\theta_1, \phi_1) \) to \( \tau(\theta_n, \phi_m) \) described by the following process: \( t \in T \) is of the form \( \tau(\theta_1, \phi_1) \triangleright_{1rw} \tau(\theta_2, \phi_2) \triangleright_{1rw} ... \triangleright_{1rw} \tau(\theta_i, \phi_i) \), with \( l_1 = 1, r_1 = 1, l_x = n, r_y = m \) and \( l_{i+1} = 1, l_i \) and \( r_{i+1} = r_i \) or \( r_{i+1} = 1 + r_i \). The independence of choice, denoted by \( cd_2 \), is defined as the rule of \( LND_{EQ} - TRS_2 \) that establishes the equality between any two different terms of \( T \). In other words, if \( x, y \in T \) and \( x \neq y \), then \( x =_{cd_2} y \) and \( y =_{cd_2} x \).

Analogously to the \( rw \)-equality, \( rw_2 \)-equality is also an equivalence relation \([32]\):

**Proposition 2.** \( rw_2 \)-equality is transitive, symmetric and reflexive.

_Proof_. Analogous to Proposition \([1]\).

### 3 A topological application of labelled natural deduction

Once we have built up all the necessary bases of computational paths to develop our work, it would be interesting to consult two proofs of the calculation of the fundamental group of the circle: The first is the mathematical proof that appears in the book of algebraic topology \([20]\) in chapter 9, section 54. The second is a proof using homotopic type theory, which is in the book in \([39]\) in chapter 8. Both cases provide the proofs of the fundamental group of the circle, but in order to obtain such success the amount of information needed is much higher and much more complex than we will propose in the next chapters.

In homotopy theory, the fundamental group is the one formed by all equivalence classes up to homotopy of paths (loops) starting from a point \( x_0 \) and also ending at \( x_0 \). Since we use computational paths as the syntactic counterpart of homotopic paths in type theory, we will use computational paths to propose some definitions that will be seen below.

Before we begin these definitions, we are going to formalize some notations that will be recurrent in the text. Consider the type \( S^1 \) (Circle), and let \( x_0 : S^1 \) be a base term of the type, and \( x_0 =_a x_0 \) be a computational path that starts and ends at the term \( x_0 \), going around the circle clockwise. We can then define this path as a base path, capable of generating any path in the circle, and denote it by \( \text{loop}_{x_0} \). However, for simplicity, we will omit the \( x_0 \), but it is implied that our loops will be made at the base point and we will denote these loops by \( \text{loop}^1 \).

Thus, the path (loop) formed by two turns based on \( x_0 \), around the circle in a clockwise direction, can be denoted by \( \text{loop}^2 \); a counterclockwise loop for \( \text{loop}^{-1} \), in general, \( \text{loop}^n \) denotes the path formed by \( n \) clockwise turns in the circle,
based on \( x_0 \), with \( n \in \mathbb{Z} \). Particularly, if \( n = 0 \), we can say that this is the homotopic path to the point and denotes it by \( \text{loop}^0 \).

Now, imagine the path formed by three clockwise turns and two counterclockwise turns. This path is different from \( \text{loop}^1 \), but it is equivalent to it, that is, we can say that it is a rewrite of the computational path \( \text{loop} \) or \( \text{loop}^1 \), so it is relevant here to define a rewrite equivalence, and we can simply denote for \( [\text{loop}^n]_{rw} \) every computational path that is equivalent, or a rewrite, of the \( \text{loop}^n \) path. Now, we can proceed with the following definitions:

**Definition 25.** Let

(i) \( A \) be a type.

(ii) \( x_0 : A \) a base point.

(iii) \( x_0 = x_0 \), be a family of generator paths with \( i \in I \).

(iv) A family of relationships between the terms paths \( \tau_j(x_0 = x_0, x_0 = x_0) \).

We can define the structure \( \Pi_1(A, x_0) \) as the set of terms \( \alpha_{x_0} \), given by finite applications of \( \tau \), \( \sigma \), and \( \rho \) in \( \alpha_i \), modulo \( rw \) equality and modulo family of identity type terms \( I_{d, \tau} \).

Since each element in \( \Pi_1(A, x_0) \) is a loop in \( x_0 \), we shall give an important definition indispensable to our work:

**Definition 26.** We can define and denote by \( [\text{loop}^n]_{rw} \)
the path naturally obtained by the application of the path-axioms \( \rho \), \( \tau \) and \( \sigma \) to the base path \( x_0 = x_0 \), where \( n \in \mathbb{N} \).

Particularly we can say:

(i) \( [\text{loop}^0]_{rw} = [\rho_{x_0}]_{rw}, n = 0 \).

(ii) \( [\text{loop}^1]_{rw} = [\text{loop}]_{rw} \).

(iii) \( [\text{loop}^n]_{rw} = \tau([\text{loop}^{n-1}]_{rw}, [\text{loop}^1]_{rw}), n > 0 \).

(iv) \( [\text{loop}^n]_{rw} = \sigma([\text{loop}^{-n}]_{rw}), -n > 0 \).

For example, we have:

a) \( \tau([\text{loop}^1]_{rw}, [\text{loop}^1]_{rw}) = [\text{loop}^2]_{rw} \)

b) \( \tau(\sigma([\text{loop}^1]_{rw}), \sigma([\text{loop}^1]_{rw})) = \sigma([\text{loop}^2]_{rw}) = [\text{loop}^{-2}]_{rw} \)

c) \( \tau(\sigma([\text{loop}^1]_{rw}), [\text{loop}^1]_{rw}) = [\rho]_{rw} \).

Here we need to provide relevant information regarding the equalities we can obtain using these paths. Consider the following examples:

\[(p_1) \quad \tau\left(\tau([\text{loop}^1]_{rw}, [\text{loop}^1]_{rw}), \sigma([\text{loop}^1]_{rw})\right) = \tau([\text{loop}^1]_{rw}, \tau([\text{loop}^1]_{rw}, \sigma([\text{loop}^1]_{rw})) \]
\[
= \tau([\text{loop}^1]_{rw}, [\rho]_{rw}) \]
\[
= [\tau\tau]_{rw} \quad [\text{loop}^1]_{rw} \]

\[(p_2) \quad \tau\left(\tau([\text{loop}^1]_{rw}, \sigma([\text{loop}^1]_{rw})), [\text{loop}^1]_{rw}\right) = \tau(\sigma([\text{loop}^1]_{rw}, [\rho]_{rw})) \]
\[
= \tau([\rho]_{rw}, [\text{loop}^1]_{rw}) \]
\[
= [\tau\rho]_{rw} \quad [\text{loop}^1]_{rw} \]

26
Notice that the paths \((p_1)\) and \((p_2)\) initially appear to be distinct paths. However, by only applying the properties of computational paths, together with the rewrite rules \((rw\text{-}rules)\), we end up with the path \([\text{loop}^1]_{rw}\) in both derivations. So we can affirm that:

By \((p_1)\),

\[
\tau\left(\tau([\text{loop}^1]_{rw}, [\text{loop}^1]_{rw}), \sigma([\text{loop}^1]_{rw})\right) =_{tr} [\text{loop}^1]_{rw}
\]

and by \((p_2)\),

\[
\tau\left(\tau([\text{loop}^1]_{rw}, \sigma([\text{loop}^1]_{rw})), [\text{loop}^1]_{rw}\right) =_{tr} [\text{loop}^1]_{rw}.
\]

They are said to be \(rw\text{-}equal\) to the base path \([\text{loop}^1]_{rw}\) because they can be rewritten to \([\text{loop}^1]_{rw}\) after the \(rw\text{-}rules\) are applied. Therefore, it can be said that these paths are in the same equivalence class as \([\text{loop}^1]_{rw}\) and thus, they are equal up to \(rw\text{-}equality\).

### 3.1 Fundamental group of the circle

**Definition 27** (The circle \(S^1\)). The circle is the type generated by:

1. A base point - \(x_0 : S^1\)
2. A base computational path - \(x_0 = x_0 : S^1\).

The first thing one should notice is that this definition does not use only the points of the type \(S^1\), but also a base computational path called \(\text{loop}\) between those points. That is why it is called a higher inductive type \([39]\). Our approach differs from the one developed in the HTT book \([59]\) in the fact that we do not need to simulate the path-space between those points, since we add computational paths to the syntax of the theory.

In Martin-Löf’s type theory, the existence of those additional paths emerges from establishing that the paths should be freely generated by the constructors \([39]\). In our approach, we do not need to appeal to this kind of argument, since all paths naturally emerge from direct applications of the axioms and the inference rules which define the theory of equality. We proceed with the following definition:

**Definition 28.** In \(S^1\), we define the following canonical loops (canonical paths):

1. \([\text{loop}^0]_{rw} = [\rho_{x_0}]_{rw}, n = 0\)
2. \([\text{loop}^1]_{rw} = [\text{loop}]_{rw}, n = 1\).
3. \([\text{loop}^n]_{rw} = \sigma([\text{loop}^n-1]_{rw}, [\text{loop}^1]_{rw}), n < 0\).
4. \([\text{loop}^n]_{rw} = \tau([\text{loop}^{n-1}]_{rw}, [\text{loop}^1]_{rw}), n > 0\).

**Lemma 1.** All paths in \(S^1\) are \(rw\text{-}equal\) to a path \([\text{loop}^n]_{rw}\), for some \(n \in \mathbb{N}\).

**Proof.** Let \(\varphi\) be a computational path in \(S^1\).

\[I.\] If \(\varphi = \rho\):

1. \(\varphi = [\text{loop}^0]_{rw}, n = 0\).
2. \(\varphi = \sigma([\text{loop}^n]_{rw}) = \sigma(\sigma([\text{loop}^n]_{rw})) = [\text{loop}^{-n}]_{rw} = \rho, n = 0\).
3. \(\varphi = \tau([\text{loop}^m]_{rw}, [\text{loop}^n]_{rw}) = \rho, if m + n = 0\). Therefore,

\[\varphi = \tau([\text{loop}^m]_{rw}, [\text{loop}^n]_{rw}) \stackrel{tr}{=} \tau([\text{loop}^n]_{rw}, [\text{loop}^n]_{rw}) \stackrel{tr}{=} \tau([\text{loop}^n]_{rw}, \sigma([\text{loop}^n]_{rw}) \stackrel{tr}{=} \rho.\]

\[II.\] If \(\varphi = \sigma([\text{loop}^n])\):

1. For \(n = 0\) we have \(\varphi = \sigma([\text{loop}^0]_{rw}) = \rho\).
2. Suppose true for \(n = k\) that every path in \(S^1\) is \(rw\text{-}equal\) to a path \([\text{loop}^n]_{rw}\). For \(n = k + 1\) we have:
\( \varphi = \tau([\text{loop}^{k+1}]_{rw}) \\
= \tau(\tau([\text{loop}^{k}]_{rw}, [\text{loop}^1]_{rw})) \\
= \tau(\tau(\tau([\text{loop}^{k}]_{rw}, [\text{loop}^1]_{rw})), [\text{loop}^1]_{rw}) \\
= \tau([\text{loop}^1]_{rw}, [\text{loop}^{-1}]_{rw}) \\
= [\text{loop}^{k+1}]_{rw} \).

### III. If \( \varphi = \tau([\text{loop}^{n-1}]_{rw}, [\text{loop}^1]_{rw}) \):

(i) For \( n = 0 \), we have:

\[ \varphi = \tau([\text{loop}^{-1}]_{rw}, [\text{loop}^1]_{rw}) = \tau(\tau([\text{loop}^1]_{rw}, [\text{loop}^1]_{rw}) = \rho = [\text{loop}^0]_{rw}. \]

(ii) Suppose true for \( n = k \), to \( n = k + 1 \) we have:

\[ \varphi = \tau([\text{loop}^{k+1}]_{rw}, [\text{loop}^1]_{rw}) \]

\[ = \tau([\text{loop}^k]_{rw}, \tau([\text{loop}^1]_{rw})) \]

\[ = \tau([\text{loop}^{k-1}]_{rw}, [\text{loop}^1]_{rw}) \]

\[ = [\text{loop}^{k+1}]_{rw}. \]

}\]

All paths in \( S^1 \) are \( rw \)-equal to a path \( [\text{loop}^n]_{rw} \), for some \( n \in \mathbb{N} \).

**Lemma 2.** All paths \( [\text{loop}^n]_{rw} \) in \( S^1 \) can be expressed in terms of \( \rho, \tau, \sigma \) and their applications, starting from the base path \( [\text{loop}^1]_{rw} \).

**Proof.** For the base case \( [\rho]_{rw} \), it is trivially true, since we define it as being equal to \([\text{loop}^0]_{rw}\). From \([\rho]_{rw}\), one can construct more complex paths by composing with \([\text{loop}^1]_{rw} \) or \( \sigma([\text{loop}^1]_{rw}) \) at each step. Concatenating the paths we obtain:

(i) A path of the form \([\rho]_{rw}\) concatenated with \([\text{loop}^1]_{rw}\):

\[ [\rho]_{rw} \circ [\text{loop}^1]_{rw} = \tau([\text{loop}^1]_{rw}, [\rho]_{rw}) = [\text{loop}^1]_{rw}. \]

(ii) A path of the form \([\rho]_{rw}\) concatenated with \( \sigma([\text{loop}^1]_{rw})\):

\[ [\rho]_{rw} \circ \sigma([\text{loop}^1]_{rw}) = \tau(\sigma([\text{loop}^1]_{rw}), [\rho]_{rw}) = \sigma([\text{loop}^1]_{rw}) = [\text{loop}^{-1}]_{rw}. \]

(iii) A path of the form \([\text{loop}^{n}]_{rw}\) concatenated with \([\text{loop}^1]_{rw}\):

\[ [\text{loop}^n]_{rw} \circ [\text{loop}^1]_{rw} = \tau([\text{loop}^1]_{rw}, [\text{loop}^n]_{rw}) = [\text{loop}^{n+1}]_{rw}. \]

(iv) A path of the form \([\text{loop}^{n}]_{rw}\) concatenated with \( \sigma([\text{loop}^1]_{rw})\):

\[ [\text{loop}^n]_{rw} \circ \sigma([\text{loop}^1]_{rw}) = \tau(\sigma([\text{loop}^1]_{rw}), [\text{loop}^n]_{rw}) \]

\[ = \tau(\tau(\sigma([\text{loop}^1]_{rw}), [\text{loop}^1]_{rw}), [\text{loop}^{n-1}]_{rw}) \]

\[ = [\text{loop}^{n+1}]_{rw}. \]
(v) A path of the form $[\text{loop}^{-n}]_{rw}$ concatenated with $[\text{loop}^1]_{rw}$:

$$[\text{loop}^{-n}]_{rw} \circ [\text{loop}^1]_{rw} = \tau([\text{loop}^1]_{rw}, [\text{loop}^{-n}]_{rw})$$

$$= \tau([\text{loop}^1]_{rw}, \tau(\sigma([\text{loop}^1]_{rw}), [\text{loop}^{-n-1}]_{rw}))$$

$$= \tau(\tau([\text{loop}^1]_{rw}, \sigma([\text{loop}^1]_{rw})), [\text{loop}^{-n-1}]_{rw})$$

$$= \tau([\rho]_{rw}, [\text{loop}^{-n-1}]_{rw})$$

$$= tl_{rr} [\text{loop}^{-n-1}]_{rw}.$$

(vi) a path of the form $[\text{loop}^{-n}]_{rw}$ concatenated with $\sigma([\text{loop}^1]_{rw})$:

$$[\text{loop}^{-n}]_{rw} \circ \sigma([\text{loop}^1]_{rw}) = \tau(\sigma([\text{loop}^1]_{rw}), [\text{loop}^{-n}]_{rw})$$

$$= \tau(\sigma([\text{loop}^1]_{rw}), \tau([\text{loop}^1]_{rw}, [\text{loop}^{-n+1}]_{rw}))$$

$$= \tau(\tau([\text{loop}^1]_{rw}, [\text{loop}^1]_{rw}), [\text{loop}^{-n+1}]_{rw})$$

$$= \tau([\rho]_{rw}, [\text{loop}^{-n+1}]_{rw})$$

$$= tl_{rr} [\text{loop}^{-n+1}]_{rw}.$$

For simplicity, we will denote by $x_0 = x_0$ whenever we refer to a computational path $r$ generated by $\rho, \sigma$ and $\tau$.

**Proposition 3.** $\Pi_1(S^1, x_0)$ provided with operations $\rho, \sigma, \tau$ is a group.

**Proof.** Given any $x_0 \cong x_0 : S^1$ and $x_0 \cong x_0 : S^1$, we need to check the group conditions:

(i) **Closure:** Given $x_0 \cong x_0 : S^1$ and $x_0 \cong x_0 : S^1$, $r \circ s$ must be a member of the group. Indeed, $r \circ s = \tau(s, r)$ is a computational path $x_0 \cong x_0 : S^1$.

(ii) **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:

$$\sigma(r) \circ r = \tau(r, \sigma(r)) \cong \rho$$

$$r \circ \sigma(r) = \tau(\sigma(r), r) \cong \rho$$

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

(iii) **Identity:** We use the path $x_0 = x_0 : S^1$ as the identity. Indeed, we have:

$$r \circ \rho = \tau(\rho, r) \cong r$$

$$\rho \circ r = \tau(r, \rho) \cong r.$$

(iv) **Associativity:** Given any member of the group $x_0 \cong x_0 : S^1$, $x_0 \cong x_0$ and $x_0 \cong x_0$, we want that $r \circ (s \circ t) = (r \circ s) \circ t$:

$$r \circ (s \circ t) = \tau(\tau(s), r) \cong \tau(t, \tau(s, r)) = (r \circ s) \circ t$$

\[29\]
Thus, all conditions have been satisfied. \[\square\]

Therefore, the structure \(\Pi_1(S^1, x_0), \rho, \sigma, \tau\) is indeed a group. We will, for simplicity, denote it in what follows for \(\Pi_1(S^1, x_0)\), and we will call it The Fundamental Group of \(S^1\).

In [39], the next theorem was proved by defining a pair of encode and decode functions. There it was necessary to simulate a path-space, and in the end the work was very laborious. Nevertheless, since our computational paths are already part of the syntax, there is no need to rely on this kind of approach to simulate a path-space. In [20] the proof of this theorem is quite laborious. By working directly with the concept of computational paths, we hope that these same calculations can be performed more simply and in such a way that is accessible to more readers.

**Theorem 4.** \(\Pi_1(S, x_0) \simeq \mathbb{Z}\)

**Proof.** Consider the application defined and denoted by:

\[
\text{toPath} : \quad \mathbb{Z} \to \Pi_1(S),
\]

\[z \to \text{toPath}(z) = \{\text{loop}_z\}_\text{rw}.
\]

(i) **toPath** is a homomorphism.

Let \(z = n + m \in \mathbb{Z}\), then:

\[
\text{toPath}(z) = \text{toPath}(n + m)
\]

\[= \{\text{loop}_{n+m}\}_\text{rw} \]

\[= \tau(\{\text{loop}_n\}_\text{rw}, \{\text{loop}_m\}_\text{rw})
\]

\[= \text{toPath}(m) \circ \text{toPath}(n).
\]

On the other hand, as \(z = m + n\) we have:

\[
\text{toPath}(z) = \text{toPath}(m + n)
\]

\[= \{\text{loop}_{m+n}\}_\text{rw} \]

\[= \tau(\{\text{loop}_m\}_\text{rw}, \{\text{loop}_n\}_\text{rw})
\]

\[= \text{toPath}(m) \circ \text{toPath}(n).
\]

Thus, \(\text{toPath}(n + m) = \text{toPath}(n) \circ \text{toPath}(m)\).

(ii) **toPath** is surjective.

By Lemma 1 as every path in \(S^1\) is \(\text{rw}\)-equal to a path \(\{\text{loop}_i\}_\text{rw}\), we have that for all paths \(\{\text{loop}_i\}_\text{rw} \in \Pi_1(S^1), \exists i \in \mathbb{Z}\), such that, \(\text{toPath}(i) = \{\text{loop}_i\}_\text{rw}\).

(iii) **Ker(toPath)** \(= \{0\}\).

Suppose there is \(z \neq 0 \in \mathbb{Z}\), such that \(z \in \text{Ker(toPath)}\). Thus,

\[
\text{toPath}(z) = \text{toPath}(z + 0) \overset{\text{hom}}{=} \text{toPath}(z) \circ \text{toPath}(0) = \tau(\rho, \{\text{loop}_z\}_\text{rw}) \overset{\text{Ker}}{=} \rho.
\]

If \(\tau(\rho, \alpha) = \rho\), by \(\text{rw}\)-rule \(\triangleright\) we have, \(\alpha = \sigma(\rho) \Rightarrow z = 0 \rightarrow_{\triangleright}\). Therefore,

\[
\text{Ker(toPath)} = \{0\}.
\]

As **toPath** is a surjective homomorphism with \(\text{Ker(toPath)} = \{0\}\), then **toPath** is an isomorphism, that is, \(\Pi_1(S, x_0) \simeq \mathbb{Z}\). \[\square\]

### 3.2 Fundamental group of the torus

Consider \(T^2\) as the surface known as Torus and the point \(x_0 \in T^2\). We will prove using computational paths that the fundamental group of the torus is isomorphic to \(\mathbb{Z} \times \mathbb{Z}\). Here we will also use Definition 25 with some simple adaptations. We will continue to work with paths up to \(\text{rw}\)-equality.

Since the fundamental groups are obtained by analysing the loops, we will be interested in working with **loops** that cannot be homotopic to base point \(x_0\), like **loops** \(\alpha\) and \(\beta\). These loops will be the generators of \(T^2\), as shown in Figure 1, so we can give them
a special definition for both. For simplicity, we will make two brief definitions to distinguish how the $\alpha$ $\beta$ loops traverse the surface of Figure 1, though this does not alter the formal definition of the loops in any way. These definitions aim at making the evidence visually clearer for the reader, avoiding very visually cluttered expressions, thus enabling a better understanding.

**Definition 29 (vertical loop).** We define and denote by

$$\alpha^n = [\text{loop}_v^n]_{rw}$$

the path that passes through the inner part of $T^2$ in the vertical direction, naturally obtained by applications of the path-axioms $\rho$, $\tau$ and $\sigma$ to the base path $x_0 = \alpha x_0$, where $n \in \mathbb{Z}$. Particularly, we have

(i) $[\text{loop}_v^0]_{rw} = [\rho]_{rw} = \alpha^0, n = 0$.

(ii) $[\text{loop}_v^{n+1}]_{rw} = \tau([\text{loop}_v^n]_{rw}, [\text{loop}_h^1]_{rw}) = \alpha^{n+1}, n > 0$.

(iii) $[\text{loop}_v^n]_{rw} = \sigma([\text{loop}_v^{-n}]_{rw}) = \alpha^{-n}, n < 0$.

In Figure 1, this vertical path (loop) has the same orientation of the path denoted by $\alpha$.

**Definition 30 (horizontal loop).** We define and denote by

$$\beta^m = [\text{loop}_h^m]_{rw}$$

the path that passes through the inner part of $T^2$ in the horizontal direction, naturally obtained by applications of the path-axioms $\rho$, $\tau$ and $\sigma$ to the base path $x_0 = \beta x_0$, where $n \in \mathbb{Z}$. Particularly, we have:

(i) $[\text{loop}_h^0]_{rw} = [\rho]_{rw} = \beta^0, m = 0$.

(ii) $[\text{loop}_h^{m+1}]_{rw} = \tau([\text{loop}_h^m]_{rw}, [\text{loop}_v^1]_{rw}) = \beta^{m+1}, m > 0$.

(iii) $[\text{loop}_h^m]_{rw} = \sigma([\text{loop}_h^{-m}]_{rw}) = \beta^{-m}, m < 0$.

In Figure 1, this horizontal path (loop) has the same orientation of the path denoted by $\beta$. By Definitions 29 and 30 we can also represent the path homotopic to the constant one by: $[\rho]_{rw} = \alpha^0 \beta^0$, or $[\rho]_{rw} = \alpha^0$, or $[\rho]_{rw} = \beta^0$. For simplicity, we denote it by $\rho$.

We now give the formal definition of the torus in homotopy type theory:

**Definition 31.** The torus $T^2$ is generated by:

(i) A base point $x_0 : T^2$.

(ii) Two base paths $\alpha$ and $\beta$ such that: $x_0 = x_0 \alpha$ and $x_0 = x_0 \beta$.

(iii) One path $co$ that establishes $\beta \alpha \equiv \alpha \beta$, i.e., a term $co : Id(\beta \alpha, \alpha \beta)$.

Based on definition 31, we can establish the following definition in computational paths:
Definition 32. In $T^2$, we define the following canonical loops (canonical paths):

(i) A base point $\alpha^0\beta^0 = [\rho_x]_{rw}$.

(ii) The path $\beta^m\alpha^n = \tau(\alpha^n, \beta^m)$.

(iii) The path $\sigma(\beta^m\alpha^n) = \sigma(\tau(\alpha^n, \beta^m))$.

(iv) One path co that establishes $\tau(\alpha^n, \beta^m) = \tau(\beta^m, \alpha^n)$.

By [20], given a point $x_0 \in T^2$, the Torus can be expressed as the quotient of a square whose sides are the base paths (loops) $\alpha$ and $\beta$, as shown in Figure 2.

![Figure 2: Square Torus representation with oriented paths $\alpha$ and $\beta$](image)

Consider the following path in the figure:

$$\alpha^{-1} \circ \beta^{-1} \circ \alpha \circ \beta = \tau(\tau(\beta, \alpha), \sigma(\beta)), \sigma(\alpha).$$

**Proposition 4.** The aforementioned path is $rw$-equal to the reflexive path.

**Proof.** Indeed,

$$\alpha^{-1} \circ \beta^{-1} \circ \alpha \circ \beta = \tau(\tau(\beta, \alpha), \sigma(\beta), \sigma(\alpha))$$

$$= \tau(\tau(\beta, \tau(\alpha, \sigma(\beta))), \sigma(\alpha))$$

$$= \tau(\tau(\beta, \sigma(\beta, \alpha)), \sigma(\alpha))$$

$$= \tau(\tau(\beta, \tau(\sigma(\beta), \alpha)), \sigma(\alpha))$$

$$= \tau(\tau(\beta, \alpha), \sigma(\alpha))$$

$$= \tau(\alpha, \sigma(\alpha))$$

$$= \rho.$$

and thus,

$$\alpha^{-1} \circ \beta^{-1} \circ \alpha \circ \beta = \tau(\tau(\beta, \alpha), \sigma(\beta)), \sigma(\alpha) \equiv [\rho]_{rw}.$$
Lemma 3. All paths in $T^2$ are $\rho\tau$-equal to the path $\beta^m\alpha^n$, with $m, n \in \mathbb{Z}$.

Proof. Let $\varphi$ be a computational path in $T^2$.

- If $\varphi = \rho$ then $\varphi = \tau(\alpha^0, \beta^0) = \beta^0\alpha^0$.
- If $\varphi = \sigma(\mu) = \sigma(\beta^m\alpha^n) = \sigma(\tau(\alpha^n, \beta^m)) = \sigma(\beta^m, \sigma(\alpha^n)) = \tau(\alpha^{-n}, \beta^{-m}) = \beta^{-m}\alpha^{-n}.

- If

$$\varphi = \tau(\mu, \omega) = \tau(\beta^m\alpha^n, \beta^s\alpha^s) = \tau(\beta^m, \tau(\alpha^n, \beta^s)) = (\beta^s \circ \alpha^s) \circ (\beta^m \circ \alpha^n) = \beta^s \circ \alpha^s \circ \beta^m \circ \alpha^n = \beta^s \circ \beta^m \circ \alpha^s \circ \alpha^n = \beta^{m+s} \alpha^{n+s}$$

Lemma 4. All paths $\beta^m\alpha^n$ in $T^2$ can be expressed in terms of $\rho, \tau, \sigma$ and their applications, starting from the base paths $\alpha^1 = [\text{loop}_v]_{\rho\tau}$ and $\beta^1 = [\text{loop}_h]_{\rho\tau}$.

Proof. Consider the following cases

(i) Base case: $\beta^0\alpha^0 = \rho$.
(ii) $\rho \circ \alpha = \tau(\alpha, \rho) \equiv \alpha = \beta^0\alpha^1$.
(iii) $\rho \circ \beta = \tau(\beta, \rho) \equiv \beta \equiv \beta^1\alpha^0$.
(iv) $\rho \circ \alpha^{-1} = \tau(\sigma(\alpha), \rho) \equiv \sigma(\alpha) = \beta^0\alpha^{-1}$.
(v) $\rho \circ \beta^{-1} = \tau(\sigma(\beta), \rho) \equiv \sigma(\beta) = \beta^{-1}\alpha^0$.

Assuming, by the induction hypothesis that every path in $T^2$ is $\rho\tau$-equal to $\beta^m\alpha^n$, we have:

1. $\rho \circ \beta^m\alpha^n = \tau(\beta^m\alpha^n, \rho) = \beta^m\alpha^n$.
2. $\alpha \circ \beta^m\alpha^n = \alpha \circ \alpha^n = \alpha^{n+1} \beta^m = \beta^m\alpha^{n+1}$.
3. $\beta \circ \beta^m\alpha^n = \beta^{n+1} \alpha^n$.
4. $\beta^{-1} \circ \beta^m\alpha^n = (\beta^{-1} \circ (\beta \circ \beta^{-1}))\alpha^n = ((\beta^{-1} \circ \beta) \circ \beta^{-1})\alpha^n = (\rho \circ \beta^{-1})\alpha^n = \beta^{-1}\alpha^n$.
5. $\alpha^{-1} \circ \beta^m\alpha^n = \alpha^{-1} \circ \alpha^n = (\alpha^{-1} \circ (\alpha \circ \alpha^{-1}))\beta^m = ((\alpha^{-1} \circ \alpha) \circ \alpha^{-1})\beta^m = (\rho \circ \alpha^{-1})\beta^m = \alpha^{-1} \beta^m = \alpha^{-1}\beta^m = \beta^{-1}\alpha^n$.

So all paths in $T^2$ are $\rho\tau$-equal to $\beta^m\alpha^n$.

Proposition 5. $\Pi_1(T^2, x_0)$ provided with operations $\rho, \sigma, \tau$ is a group.

Proof.

(i) Closure:
\[
\begin{align*}
X_0 & \equiv X_0 \\
& \equiv X_0 \\
\therefore X_0 & \equiv X_0 \\
\end{align*}
\]

But,
\[
\tau(\beta^u\alpha^v, \beta^r\alpha^s) = (\beta^r\alpha^s) \circ (\beta^u\alpha^v) = \beta^u\alpha^v \beta^r\alpha^s = \beta^u \beta^r \alpha^v \alpha^s = \beta^{u+r} \alpha^{v+s}.
\]

(ii) Inverse:
\[
\begin{align*}
X_0 & \equiv X_0 \\
& \equiv X_0 \\
\therefore X_0 & \equiv X_0 \\
\end{align*}
\]

But,
\[
\tau(\beta^m\alpha^n, \sigma(\beta^m\alpha^n)) = (\sigma(\beta^m\alpha^n)) \circ (\beta^m\alpha^n) = \sigma(\beta^m\alpha^n) \beta^m\alpha^n = \sigma(\beta^n) \beta^m\sigma(\alpha^n) = \beta^m \beta^r \sigma(\alpha^n) = \beta^m \beta^r \alpha^n = \beta^{m+r} \alpha^n
\]

On the other hand, we have:
\[
\begin{align*}
X_0 & \equiv X_0 \\
& \equiv X_0 \\
\therefore X_0 & \equiv X_0 \\
\end{align*}
\]

But,
\[
\tau(\sigma(\beta^n)\sigma(\alpha^n), \beta^m\alpha^n) = (\sigma(\beta^n)\sigma(\alpha^n)) \circ (\beta^m\alpha^n) = \beta^m \sigma(\beta^n)\sigma(\alpha^n) = \beta^m \alpha^n \sigma(\beta^n)\sigma(\alpha^n) = \beta^m \alpha^n \sigma(\beta^n)\alpha^n = \beta^{m+n} \sigma(\alpha^n) = \beta^{m+n} \alpha^n
\]

(iii) Identity:
\[
\begin{align*}
X_0 & \equiv X_0 \\
& \equiv X_0 \\
\therefore X_0 & \equiv X_0 \\
\end{align*}
\]

But,
\[
\tau(\beta^m\alpha^n, \rho x_0) = (\rho x_0) \circ (\beta^m\alpha^n) = \rho x_0 \beta^m\alpha^n = \beta^m\alpha^n
\]
and so
\[ \tau(\beta^m \alpha^n, \rho_{x_0}) = \beta^m \alpha^n. \]

On the other hand, we have:
\[
\begin{array}{c}
\frac{x_0 = x_0}{x_0 = x_0} \\
\tau(\rho_{x_0}, \beta^m \alpha^n) = x_0
\end{array}
\]

But,
\[
\tau(\rho_{x_0}, \beta^m \alpha^n) = (\beta^m \alpha^n) \circ (\rho_{x_0}) = \beta^m \alpha^n \rho_{x_0} = \beta^m \alpha^n
\]

and so
\[
\tau(\rho_{x_0}, \beta^m \alpha^n) = \beta^m \alpha^n.
\]

(iv) **Associativity:**
\[
\begin{array}{c}
x_0 = x_0 \\
\tau(\beta^m \alpha^n, \beta^i \alpha^j) = x_0 \\
\tau(\beta^j \alpha^s, \beta^r \alpha^t) = x_0
\end{array}
\]

But,
\[
\tau(\beta^m \alpha^n, \beta^i \alpha^j) = (\beta^j \alpha^s) \circ (\beta^i \alpha^j \circ \beta^m \alpha^n) = (\beta^j \alpha^s) \circ (\beta^i \alpha^j \beta^m \alpha^n) = \beta^j \alpha^s \beta^i \alpha^j \beta^m \alpha^n.
\]

On the other hand, we have:
\[
\begin{array}{c}
x_0 = x_0 \\
\tau(\beta^m \alpha^n, \tau(\beta^j \alpha^s, \beta^r \alpha^t)) = x_0 \\
\end{array}
\]

But,
\[
\tau(\beta^m \alpha^n, \tau(\beta^j \alpha^s, \beta^r \alpha^t)) = \tau(\beta^j \alpha^s, \beta^r \alpha^t) \circ (\beta^m \alpha^n) = (\beta^j \alpha^s \circ \beta^j \alpha^s) \circ (\beta^m \alpha^n) = (\beta^j \alpha^s \beta^j \alpha^s) \circ (\beta^m \alpha^n) = \beta^j \alpha^s \beta^j \alpha^s \beta^m \alpha^n.
\]
Therefore, the structure \( (\Pi_1(T^2, x_0), \rho, \sigma, \tau) \) is indeed a group. We will, for simplicity, denote it in what follows for \( \Pi_1(T^2, x_0) \), and we will call it The Fundamental Group of \( T^2 \).

\[ \square \]

**Theorem 5.** \( \Pi_1(T^2, x_0) \cong \mathbb{Z} \times \mathbb{Z} \).

**Proof.** Consider the map:

\[
\text{toPath}^2 : \mathbb{Z} \times \mathbb{Z} \rightarrow \Pi_1(T^2, x_0) \\
(m, n) \rightarrow \beta^m \alpha^n.
\]

i) toPath\(^2\) is a homomorphism.

Let \((m_1 + m_2, n_1 + n_2) \in \mathbb{Z} \times \mathbb{Z}\), then:

\[
toPath^2(m_1 + m_2, n_1 + n_2) = \beta^{m_1 + m_2} \alpha^{n_1 + n_2} = \beta^{m_1} \beta^{m_2} \alpha^{n_1} \alpha^{n_2} = \beta^{m_1} \alpha^{n_1} \beta^{m_2} \alpha^{n_2} = toPath^2(m_1, n_1) \circ toPath^2(m_2, n_2).
\]

ii) toPath is surjective.

By Lemma 3, as every path in \( T^2 \) is \( rw \)-equal to a path \( \beta^m \alpha^n \), we have that for all paths \( \beta^i \alpha^j \in \Pi_1(T^2) \), \( \exists (i, j) \in \mathbb{Z} \times \mathbb{Z} \), such that, \( toPath(i, j) = \beta^i \alpha^j \).

iii) Ker(toPath\(^2\)) = \{(0, 0)\}.

Suppose that \((m, n) \neq (0, 0) \in \mathbb{Z} \times \mathbb{Z}\), such that \((m, n) \in \text{Ker(toPath}^2\))). Thus,

\[
toPath^2(m, n) = toPath^2(m + 0, n + 0) \overset{\text{hom}}{=} toPath^2(m, n) \circ toPath^2(0, 0) = \beta^m \alpha^n \beta^0 \alpha^0 = \tau(\beta^0 \alpha^0, \beta^m \alpha^n) = \tau(\rho, \beta^m \alpha^n) = \rho.
\]

If \( \tau(\rho, \alpha) = \rho \), by \( rw \)-rule \( \rho \) we have, \( \alpha = \sigma(\rho) \Rightarrow (m, n) = (0, 0) \rightarrow \rho \). Therefore,

\[
\text{Ker(toPath}^2\) = \{(0, 0)\}.
\]

As toPath\(^2\) is a homomorphism surjective with Ker(toPath\(^2\)) = \{(0, 0)\}, then toPath\(^2\) is an isomorphism, that is, \( \Pi_1(T^2) \cong \mathbb{Z} \times \mathbb{Z} \).

\[ \square \]

### 3.3 Fundamental group of the real projective plane

The real projective plane, denoted by \( \mathbb{R}P^2 \), is by definition the set of all straight lines that pass through the origin of space \( \mathbb{R}^3 \). We can define each of these lines by a position vector \( v_r \), with \( \|v_r\| \neq 0 \), so we have that \( \mathbb{R}P^2 \) is a quotient space of \( \mathbb{R}^3 - (0, 0, 0) \) under the equivalence relation \( v_r \sim \lambda v_r \) for scalars \( \lambda \neq 0 \). If we impose the condition that the vectors \( \|v_r\| = 1 \) then \( \mathbb{R}P^2 \) is a quotient space \( S^2 \) under the equivalence relation \( v_r \sim -v_r \), the sphere with antipodal points identified, where \( v_r \) is position vector.

Let \( [v_r] = [x, y, z] \), where \([x, y, z][v_r] = (x, y, z), -v_r = (-x, -y, -z)\) with \( z \neq 0 \). This is equivalent to saying that \( \mathbb{R}P^2 \) is the quotient space of an upper hemisphere \( S^2 \) with antipodal points of \( \partial S^2 \) identified, as shown in Figure 3.

Let us then map it on the unit disk through the following map \([x, y, z] \rightarrow (x, y, 0)\), as follows in Figure 4.
A Topological Application of Labelled Natural Deduction

**Figure 3:** $P$ and $\tilde{P}$ are antipodal points in $\partial D^2$.

**Figure 4:** Mapping Projection on the unit disk on the $xy$ plane.

This way we have that $\mathbb{RP}^2$ is a quotient space of $D$ with antipodal points of $\partial D$ identified. Therefore we can study the fundamental group of $\mathbb{RP}^2$ from the disk shown on the right side of **Figure 4**.

We denote by $\alpha$ any loop that connects the identified antipodal points, so we can consider $\alpha$ as a loop (as follows in **Figure 5**) and any other loop that connects the identified antipodal points is homotopic to $\alpha$. Note that $\forall Q \in D$, any loop based on $Q$ is homotopic to the point, and it is not in our interest to study those.

**Figure 5:** loop $\alpha$.

Since we can represent the real projective plane $\mathbb{RP}^2$ as a disk $D$, we can define $\mathbb{RP}^2$, homotopically, as follows:

**Definition 33.** The real projective plane $\mathbb{RP}^2$ is defined by:

(i) The types $Q : D$, such that $Q \in D$.

(ii) The pair $P, P' : \partial D$, such that: $P, P'$ are the pairs of antipodal points identified in $\partial D$.

(iii) A path $\alpha$ such that: $P = p'$.  

(iv) A path cicl that establishes $\alpha \circ \alpha = \text{cicl}$, i.e, $\text{cicl} : \text{Id}_{\mathbb{RP}^2}(\alpha \circ \alpha, \rho)$.

**Lemma 5.** All paths in $\mathbb{RP}^2$ can be expressed in terms of $\rho, \tau, \sigma$ and their applications, starting from the base paths $\rho$ or $\alpha$.

**Proof.** Consider the following base cases:

**Base case** $\varphi = \rho$:

(i) Trivial case.
Base case $\varphi = \sigma(\phi)$:

(i) $\varphi = \sigma(\alpha) = \text{cicl}\alpha$.

(ii) $\varphi = \sigma(\rho) = \text{rw}\rho$.

Base case $\varphi = \tau(\phi, \kappa)$:

(i) $\varphi = \tau(\rho, \rho) = \text{trr}\rho$.

(ii) $\varphi = \tau(\alpha, \rho) = \text{trr}\alpha$.

(iii) $\varphi = \tau(\rho, \alpha) = \text{tlr}\alpha$.

(iv) $\varphi = \tau(\alpha, \alpha) = \text{cicl}\rho$.

**Inductive case:** Assuming true for $n$, we have:

If $[\text{loop}^n]\text{rw} = [\rho]\text{rw}$, we have two possibilities for $n + 1$:

(i) $[\text{loop}^{n+1}]\text{rw} = [\text{loop}^n]\text{rw} \circ \rho = \tau(\rho, \rho) = [\rho]\text{rw}$.

(ii) $[\text{loop}^{n+1}]\text{rw} = \tau(\alpha, \rho) = [\rho]\text{rw} \circ \tau(\rho, \rho) = [\rho]\text{rw}$.

If $[\text{loop}^n]\text{rw} = \alpha$, we have two possibilities for $n + 1$:

(i) $[\text{loop}^{n+1}]\text{rw} = [\text{loop}^n]\text{rw} \circ \alpha = \alpha \circ \rho = \tau(\rho, \alpha)$.

(ii) $[\text{loop}^{n+1}]\text{rw} = [\text{loop}^n] \circ \alpha = \alpha \circ \alpha = \tau(\alpha, \alpha)$.

Thus, all paths in $\mathbb{RP}^2$ generated by $\rho$ or $\alpha$ are $\text{rw}$-equal to either $\alpha$ or $\rho$. Since we have $\alpha \circ \alpha = \tau(\alpha, \alpha) = [\rho]\text{rw}$, the term $\text{cicl}\rho$ gives us one important result: $\alpha = \sigma(\alpha)$.

**Proposition 6.** $\Pi_1(\mathbb{RP}^2)$ provided with operations $\rho, \sigma, \tau$ is a group.

**Proof.**

(i) **Closure:**

\[
\begin{align*}
P \equiv P \circ P & \equiv P \\
\tau(\alpha, \alpha) & \equiv P
\end{align*}
\]

But,

\[\alpha \circ \alpha = \tau(\alpha, \alpha) = [\rho]\text{rw} \in \Pi_1(\mathbb{RP}^2)\,.
\]

(ii) **Inverse:**

\[
\begin{align*}
P \equiv P \circ P & \equiv P \\
\tau(\alpha, \sigma(\alpha)) & \equiv P
\end{align*}
\]

But,

\[\sigma(\alpha) \circ \alpha = \tau(\alpha, \sigma(\alpha)) = [\rho]\text{rw} \in \Pi_1(\mathbb{RP}^2)\,.
\]

On the other hand, we have:
\[
\frac{P = P}{P_{\sigma(\alpha)}} \quad \frac{P = P}{P_{\alpha}}
\]

But,

\[
\alpha \circ \sigma(\alpha) = \tau(\sigma(\alpha), \alpha) = \rho \in \Pi_1(\mathbb{R}P^2).
\]

(iii) **Identity:**

\[
\frac{P = P}{P_{\rho}} \quad \frac{P = P}{P_{\alpha}}
\]

But,

\[
\rho \circ \alpha = \tau(\alpha, \rho) = \alpha \in \Pi_1(\mathbb{R}P^2).
\]

On the other hand, we have:

\[
\frac{P = P}{P_{\rho}} \quad \frac{P = P}{P_{\alpha}}
\]

But,

\[
\alpha \circ \rho = \tau(\rho, \alpha) = \alpha \in \Pi_1(\mathbb{R}P^2).
\]

(iv) **Associativity:**

\[
\frac{P = P}{P_{\tau(\alpha, \alpha)}} \quad \frac{P = P}{P_{\alpha}}
\]

But,

\[
\tau(\tau(\alpha, \alpha), \alpha) = \alpha \circ \tau(\alpha, \alpha) \\
= \alpha \circ \rho \\
= \tau(\rho, \alpha) \\
= \alpha
\]

On the other hand, we have:

\[
\frac{P = P}{P_{\tau(\alpha, \alpha)}} \quad \frac{P = P}{P_{\alpha}}
\]

39
But,

\[
\tau(\alpha, \tau(\alpha, \alpha)) = \tau(\alpha, \alpha) \circ \alpha \\
= \rho \circ \alpha \\
= \tau(\alpha, \rho) \\
= \tau(\alpha, \alpha) \circ \alpha
\]

Since \(\tau(\alpha, \alpha) = \tau(\alpha, \alpha) \circ \alpha\), it follows that associativity is valid and therefore \((\Pi_1(\mathbb{R}P^2), \circ)\) is a group generated by \(\rho\) and \(\alpha\).

**Theorem 6.** \(\Pi_1(\mathbb{R}P^2) \simeq \mathbb{Z}_2\).

**Proof.** Consider the application defined and denoted by:

\[
toPath_{\mathbb{Z}_2} : \mathbb{Z}_2 \to \Pi_1(\mathbb{R}P^2) \\
z \mapsto toPath_{\mathbb{Z}_2} = \left[\text{loop}_z\right]_{rw}.
\]

(i) \(toPath_{\mathbb{Z}_2}\) is a homomorphism.

Let \(z_1\) and \(z_2\) \(\in\) \(\mathbb{Z}_2\), then:

\[
toPath_{\mathbb{Z}_2}(z_1 + z_2) = \left[\text{loop}_{z_1 + z_2}\right]_{rw} \\
= \tau([\text{loop}^z_{z_1}]_{rw}, [\text{loop}^z_{z_2}]_{rw}) \\
= toPath_{\mathbb{Z}_2}(z_2) \circ toPath_{\mathbb{Z}_2}(z_1).
\]

On the other hand, we have:

\[
toPath_{\mathbb{Z}_2}(z_2 + z_1) = \left[\text{loop}^{z_2 + z_1}\right]_{rw} \\
= \tau([\text{loop}^z_{z_2}]_{rw}, [\text{loop}^z_{z_1}]_{rw}) \\
= toPath_{\mathbb{Z}_2}(z_1) \circ toPath_{\mathbb{Z}_2}(z_2).
\]

Thus, \(toPath_{\mathbb{Z}_2}(z_1 + z_2) = toPath_{\mathbb{Z}_2}(z_2) \circ toPath(z_1)\).

(ii) \(toPath_{\mathbb{Z}_2}\) is surjective.

By Lemma\[5\] every path in \(\mathbb{R}P^2\) is \(rw\)-equal to \(\rho\) and \(\alpha\). So given any path in \(\Pi_1(\mathbb{R}P^2)\), for \(z = 0\) and \(z = 1\) we have \(\rho = toPath_{\mathbb{Z}_2}(0)\) and \(\alpha = toPath_{\mathbb{Z}_2}(1)\), respectively.

(iii) \(\ker(toPath_{\mathbb{Z}_2}) = \{0\}\).

By Lemma\[5\] there is only one element in \(z \in \mathbb{Z}_2\) such that \(toPath_{\mathbb{Z}_2}(z) = 0\). Therefore, \(\ker(toPath_{\mathbb{Z}_2}) = \{0\}\).

\[
toInt = \begin{cases} 
\text{toInt}([\text{loop}^0]_{rw} = [\rho]_{rw}) = 0 \\
\text{toInt}([\text{loop}^1]_{rw} = \alpha) = 1 
\end{cases}
\]

Thus, the isomorphism holds.

**4 Conclusion**

Our main objective has been the calculation of the fundamental groups of many surfaces using a labelled deduction system based on the concept of computational paths (sequences of rewrites). The main advantage of this approach is that we avoid the use of more complex techniques, such as those made in algebraic topology in pure mathematics or by the method of encoding-decoding used in homotopy type theory. As a consequence, our calculations proved to be straightforward and simple. Using computational paths as our main tool, we have calculated the fundamental group of the circle, torus and projective plane. Therefore, we have shown that it is possible to use the theory of computational paths to obtain useful results in algebraic topology.

Finally, an almost natural question for our study would be: is it possible to calculate the fundamental group of the Klein bottle using the same technique? This question is going to be a guiding element to develop our future studies.
References

[1] CHENADEC, P. L. On the logic of unification. *Journal of Symbolic computation* 8(1):141–199, Elsevier, 1989.

[2] CHURCH, A. An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics* 58(2):345–363, 1936.

[3] CURRY, H. B. 1900-1982, HINDLEY, J.R. and SELDIN, J. P. In *To H. B. Curry: Essays on Combinatory Logic Lambda Calculus and Formalism*. edited by J.P. Seldin, J.R. Hindley. p.xxxi+606pp, 1980. London by Academic press.

[4] DERSHOWITZ, N. Orderings for term-rewriting systems. *Theoretical computer science* 17(3):279–301, Elsevier, 1982.

[5] GAMBINO, N. and GARNER, R. The identity type weak factorisation system. *Theoretical Computer Science* 409(1):94–109, 2008

[6] HARPER, R. Type Theory Foundations. 2012. Type Theory Foundations, Lecture at Oregon Programming Languages Summer School, Eugene, Oregon.

[7] HINDLEY, J. R. and SELDIN, J. P. *Lambda-calculus and combinators: an introduction*. 2008. Cambridge University Press.

[8] HOFMANN, M.; STREICHER, T. The groupoid model refutes uniqueness of identity proofs. In *IEEE Logic in Computer Science, 1994. LICS ’94*. Proceedings., Symposium on. [S.l.], 1994.p. 208–212

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

[10] KRAUS, N.; von RAUMER, J. Path Spaces of Higher Inductive Types in Homotopy Type Theory. *LICS’19*. DOI: 10.1109/LICS.2019.8785661 (Preliminary version: arXiv:1901.06022)

[11] LICATA, D. R. and Shulman, M. Calculating the Fundamental Group of the Circle in Homotopy Type Theory. 28th Annual *ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2013. ACM/IEEE.

[12] MARTIN-LÖF, P. An intuitionistic theory of types: predicative part, in H. E. Rose and J.C.Shepherdson (eds.) *Logic Colloquium ’73*, Vol. 80 of Studies in Logic and The Foundations of Mathematics, pp 73–118, North-Holland, Amsterdam,1975. p.viii+513pp, Proceedings of the Colloquium held in Bristol,UK,1973.

[13] MARTIN-LÖF, P. Constructive Mathematics and Computer Programming, in L. J. Cohen, J. Los, H.Pfeiffer, and K.-P. Podewski (eds.) *Logic, Methodology and Philosophy of Science VI*. Series Studies in Logic and The Foundations of Mathematics,pp 153—175, North-Holland, Amsterdam. p.xiii+738pp,1992. Proceedings of the International Congress held in Hannover, August 22–29 1979.

[14] MARTIN-LÖF, P. Intuitionistic Type Theory. Series Studies in Proof Theory. Bibliopolis Naples, iv+91pp, 1980. Notes by Giovanni Sambin of a series of lectures given in Padova.

[15] MARTIN-LÖF, P. Truth of a proposition, evidence of a judgement, validity of a proof. *Synthese* 73:407–420, 1987.

[16] MARTINEZ-RIVILLAS, D.O. and de QUEIROZ, R. J. G. B. ∞-Groupoid Generated by an Arbitrary Topological λ-Model. *Logic Journal of the IGPL* 2021 (to appear). https://doi.org/10.1093/jigpal/jzab015 (also arXiv:1906.05729)

[17] MARTINEZ-RIVILLAS, D.O. and de QUEIROZ, R. J. G. B. Towards a Homotopy Domain Theory. arXiv:2007.15082

[18] MUNKRES, J. R. Topology (2nd Edition), Chapter 9, Topic 54 - The fundamental group of circle, Pearson, 2000. Institute for Advanced Study.

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

[20] de OLIVEIRA, A. G. and de QUEIROZ, R. J. G. B. A New Basic Set of Proof Transformations. In We Will Show Them! *Essays in Honour of Dov Gabbay*. S. Artemov, H. Barringer, A. Garcez, L. Lamb and J. Woods (eds.), pp.499–528. 2005. College Publications, London, ISBN 1904987125.

[21] de OLIVEIRA, A. G. and de QUEIROZ, R. J. G. B. A normalization procedure for the equational fragment of labelled natural deduction. *Logic Journal of IGPL* 7(2):173–215. March 1999. https://doi.org/10.1093/jigpal/7.2.173

[22] de QUEIROZ, R. J. G. B. and GABBAY, D. M. Equality in Labelled Deductive Systems and the Functional Interpretation of Propositional Equality. *9th Amsterdam Colloquium*, p.547-565, 1994. ILLC/Department of Philosophy, University of Amsterdam.

[23] de QUEIROZ, R. J. G. B. and GABBAY, D. M. The Functional Interpretation of the Existential Quantifier. *Bulletin of the Interest Group in Pure and Applied Logics* 3(2–3):243–290, 1995. https://doi.org/10.1093/jigpal/3.2-3.243 (Abstract in JSL 58(2):753–754, 1993. Presented at Logic Colloquium ’91, Uppsala, August 9–16.)

[24] de QUEIROZ, R. J. G. B. and Gabbay,D. M. Labelled Natural Deduction. In *Logic, Language and Reasoning. Essays in Honor of Dov Gabbay’s 50th Anniversary*, H.J. Ohlbach and U. Reyle (eds.). Kluwer Academic Publishers, June 1999. pp. 173—250.
[27] de QUEIROZ, R. J. G. B. and de OLIVEIRA, A. G. and Ramos, A. F. 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).

[28] de QUEIROZ, R. J. G. B. and de OLIVEIRA, A. G. Term rewriting systems with labelled deductive systems. *Proceedings of Brazilian Symposium on Artificial Intelligence (SBIA’94)* p.59-74, 1994.

[29] de QUEIROZ, R. J. G. B. de OLIVEIRA, A. G. Natural deduction for equality: The missing entity. *Advances in Natural Deduction - A Celebration of Dag Prawitz’s Work*, p.63-91. Springer, 2014.

[30] de QUEIROZ, R. J. G. B. and de OLIVEIRA, A. G. Propositional Equality, Identity Types and Reversible Rewriting Sequences as Homotopies. Talk given at Workshop of Logic, Universidade Federal do Ceará, Fortaleza, CE, 2014.

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

[32] RAMOS, A.F. and de QUEIROZ, R. J. G. B. and de OLIVEIRA, A. G. On the identity type as the type of computational paths. *Logic Journal of the IGPL* 25 (4), p.562-584, 2017.

[33] RAMOS, A.F. and de QUEIROZ, R. J. G. B. and de OLIVEIRA, A. G. Sequences of Rewrites: A Categorical Interpretation. *arXiv:1412.2105*

[34] RAMOS, A.F. and de QUEIROZ, R. J. G. B. and de OLIVEIRA, A. G. On the Groupoid Model of Computational Paths. *arXiv:1506.02271*

[35] RAMOS, A.F. and de QUEIROZ, R. J. G. B. and de OLIVEIRA, A. G. On the Identity Type as the Type of Computational Paths. *arXiv:1504.04759*

[36] RAMOS, A.F. Explicit Computational Paths in Type Theory. PhD Thesis, 2018. Depto. de Informática, Universidade Federal de Pernambuco, Recife, Brazil, August, 2018.

[37] RAMOS, A.F. and de QUEIROZ, R. J. G. B. and de OLIVEIRA, A. G. and de VERAS, T. M. L. Explicit Computational Paths. *South American Journal of Logic* Vol. 4, n. 2, pp. 441–484, 2018 ISSN: 2446-6719.

[38] RAMOS, A.F. and de QUEIROZ, R. J. G. B. and de OLIVEIRA, A. G. and de VERAS, T. M. L. On the Use of Computational Paths in Path Spaces of Homotopy Type Theory. *arXiv:1804.01413*

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

[40] VERAS, T. M. L., Arthur F. Ramos, Ruy J. G. B. de Queiroz, Thiago D. O. Silva, Anjolina G. de Oliveira. Computational Paths - A Weak Groupoid. *arXiv:2007.07769*

[41] VERAS, T. M. L., RAMOS, A.F., de QUEIROZ, R. J. G. B., de OLIVEIRA, A. G. An alternative approach to the calculation of fundamental groups based on labeled natural deduction. *arXiv:1906.09107*

[42] VERAS, T. M. L., RAMOS, A.F., de QUEIROZ, R. J. G. B., de OLIVEIRA, A. G. On the Calculation of Fundamental Groups in Homotopy Type Theory by Means of Computational Paths. *arXiv:1804.01413*

[43] VOEVODSKY, V. Notes on homotopy λ-calculus. (Started Jan. 18, Feb. 11, 2006.) https://www.math.ias.edu/~vladimir/Site3/Univalent_Foundations_files/homotopy_lambda_calculus_3.pdf

[44] VOEVODSKY, V. Univalent Foundations and Set Theory, 2014. Lecture at IAS, Princeton, New Jersey, Mar 2014.