On the identity type as the type of computational paths

ARTHUR F. RAMOS*, RUY J. G. B. DE QUEIROZ** and ANJOLINA G. DE OLIVEIRA†, *Centro de Informática, Universidade Federal de Pernambuco, Recife, Brazil.*

Abstract

We present a way of formalizing the intensional identity type based on the notion of computational paths which will be taken to be proofs of propositional equality, and thus terms of the identity type. This approach results in an elimination rule which constructs some relevant basic types of Martin-Löf's Intensional Type Theory in a simpler way. To make this point clear we construct terms of these types using our proposed elimination rule. We also show that one of the properties of Martin-Löf's original identity type is present on our formulation of the identity type of computational paths. We are referring to the fact that the identity type induces a groupoid structure, as proposed by Hofmann and Streicher in 1994. Using categorical semantics, we show that computational paths induce a groupoid structure too. It is further shown that computational paths induce higher categorical structures. Using our groupoid structure, we also show that our approach refutes the uniqueness of identity proofs. This is on par with the same result obtained by Hofmann and Streicher in 1995 for the original identity type.

Keywords: Identity type, computational paths, equality proofs, path-based constructions, type theory, groupoid model, term rewriting systems, category theory, higher categorical structures, uniqueness of identity proofs.

1 Introduction

One interesting peculiarity of Martin-Löf's Intensional Type Theory is the existence of two distinct kinds of equalities between terms of the same type. The first one is originated by the fact that equality can be seen as a type. The second one is originated by the fact that two terms can be equal by definition.

The treatment of an equality as a type gives rise to an extremely interesting type known as identity type. The idea is that, given terms a, b of a type A, one may form the type whose elements are proofs that a and b are equal elements of type A. This type is represented by $Id_A(a, b)$. A term $p : Id_A(a, b)$ makes up for the *grounds* [17] (or proof) that establishes that a is indeed equal to b. We say that a is *propositionally* equal to b.

The second kind of equality is called definitional and is denoted by $\equiv$. It occurs when two terms are equal by definition, i.e., there is no need for an evidence or a proof to establish the equality. A classic example, given in [18], is to consider any function definition, e,g., $f : \mathbb{N} \to \mathbb{N}$ defined by $f(x) \equiv x^2$. For $x = 3$, we have, by definition, that $f(x) \equiv 3^2$. We are unable to conclude that $f(3) \equiv 9$ though. The problem is that, to conclude that $3^2 = 9$, we need additional evidence. We need a way to compute 3^2, i.e., we need to use exponentials (or multiplications) to conclude that 3^2 is equal to 9. That way, we can only prove that 3^2 is propositionally equal to 9, i.e., there is a $p : Id_{\mathbb{N}}(3^2, 9)$.

*E-mail: afr@cin.ufpe.br
**E-mail: ruy@cin.ufpe.br
†E-mail: ago@cin.ufpe.br

doi:10.1093/jigpal/jzx015 Advance Access published 26 June 2017

Between those two kind of equalities, the propositional one is, without a doubt, the most interesting one. This claim is based on the fact that many interesting results have been achieved using the identity type. One of these was the discovery of the Univalent Models in 2005 by Vladimir Voevodsky [20]. A groundbreaking result has arisen from Voevodsky's work: the connection between type theory and homotopy theory. The intuitive connection is simple: a term $a:A$ can be considered as a point of the space A and $p:Id_A(a,b)$ is a homotopy path between points $a,b \in A$ [18]. This has given rise to a whole new area of research, known as Homotopy Type Theory. It leads to a new perspective on the study of equality, as expressed by Voevodsky in a recent talk in *The Paul Bernays Lectures* (September 2014, Zürich): equality (for abstract sets) should be looked at as a *structure* rather than as a *relation*.

Motivated by the fact that the identity type has given rise to such interesting concepts, we have been engaged in a process of revisiting the construction of the intensional identity type, as originally proposed by Martin-Löf. Although beautifully defined, we have noticed that proofs that uses the identity type can be sometimes a little too complex. The elimination rule of the intensional identity type encapsulates lots of information, sometimes making too troublesome the process of finding the reason that builds the correct type.

Inspired by the path-based approach of the homotopy interpretation, we believe that a similar approach can be used to define the identity type in type theory. To achieve that, we have been using a notion of *computational paths*. The interpretation will be similar to the homotopy one: a term $p:Id_A(a,b)$ will be a computational path between terms $a,b:A$, and such path will be the result of a sequence of rewrites. In the sequel, we shall define formally the concept of a computational path. The main idea, 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 [7].

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 $Id_A(a,b)$. If this type is inhabited by any $p,q:Id_A(a,b)$, then we have type $Id_{Id_A(a,b)}(p,q)$. If the latter type is inhabited, we have a higher equality between p and q [9]. This concept is also present in computational paths. One can show the equality between two computational paths s and t by constructing a third one between s and t. We show in the sequel a system of rules used to establish equalities between computational paths [2]. Then, we show that these higher equalities go up to the infinity, forming a ∞-globular-set. We also show that computational paths naturally induce a structure known as groupoid. We also go a step further, showing that computational paths are capable of inducing a higher groupoid structure.

2 Computational paths

Since computational path is a generic term, it is important to emphasize the fact that we are using the term computational path in the sense defined by [6]. A computational path is based on the idea that it is possible to formally define when two computational objects $a,b:A$ are equal. These two objects are equal if one can reach b from a applying a sequence of axioms or rules. This sequence of operations forms a path. Since it is between two computational objects, it is said that this path is a computational one. Also, an application of an axiom or a rule transforms (or rewrite) an term in another. For that reason, a computational path is also known as a sequence of rewrites. Nevertheless,

before we define formally a computational path, we can take a look at one famous equality theory, the $\lambda\beta\eta-$*equality* [10]:

DEFINITION 2.1
The $\lambda\beta\eta$*-equality* is composed by the following axioms:

(α) $\lambda x.M = \lambda y.M[y/x]$ if $y \notin FV(M)$;
(β) $(\lambda x.M)N = M[N/x]$;
(ρ) $M = M$;
(η) $(\lambda x.Mx) = M$ $(x \notin FV(M))$.

And the following rules of inference:

$$(\mu)\quad \frac{M = M'}{NM = NM'} \qquad (\tau)\ \frac{M = N \qquad N = P}{M = P}$$

$$(\nu)\quad \frac{M = M'}{MN = M'N} \qquad (\sigma)\ \frac{M = N}{N = M}$$

$$(\xi)\quad \frac{M = M'}{\lambda x.M = \lambda x.M'}$$

DEFINITION 2.2 ([10])
P is β-equal or β-convertible to Q (notation $P =_\beta Q$) iff Q is obtained from P by a finite (perhaps empty) series of β-contractions and reversed β-contractions and changes of bound variables. That is, $P =_\beta Q$ Iff **there exist** $P_0, \ldots, P_n$ $(n \geq 0)$ such that $P_0 \equiv P$, $P_n \equiv Q$, $(\forall i \leq n-1)(P_i \rhd_{1\beta} P_{i+1}$ or $P_{i+1} \rhd_{1\beta} P_i$ or $P_i \equiv_\alpha P_{i+1})$.

(NB: equality with an **existential** force, which will show in the proof rules for the identity type.)
The same happens with $\lambda\beta\eta$-equality:

DEFINITION 2.3 ($\lambda\beta\eta$-equality [10])
The equality-relation determined by the theory $\lambda\beta\eta$ is called $=_{\beta\eta}$; i.e., we define

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

EXAMPLE 2.4
Take the term $M \equiv (\lambda x.(\lambda y.yx)(\lambda w.zw))v$. Then, it is $\beta\eta$-equal to $N \equiv zv$ because of the sequence:
$(\lambda x.(\lambda y.yx)(\lambda w.zw))v$, $(\lambda x.(\lambda y.yx)z)v$, $(\lambda y.yv)z$, zv
which starts from M and ends with N, and each member of the sequence is obtained via 1-step β- or η-contraction of a previous term in the sequence. To take this sequence into a *path*, one has to apply transitivity twice, as we do in the example below.

EXAMPLE 2.5
The term $M \equiv (\lambda x.(\lambda y.yx)(\lambda w.zw))v$ is $\beta\eta$-equal to $N \equiv zv$ because of the sequence:
$(\lambda x.(\lambda y.yx)(\lambda w.zw))v$, $(\lambda x.(\lambda y.yx)z)v$, $(\lambda y.yv)z$, zv.
Now, taking this sequence into a path leads us to the following:
The first is equal to the second based on the grounds:
$\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v)$.
The second is equal to the third based on the grounds:
$\beta((\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z)$.

Now, the first is equal to the third based on the grounds:
$\tau(\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v,(\lambda x.(\lambda y.yx)z)v),\beta((\lambda x.(\lambda y.yx)z)v,(\lambda y.yv)z))$.
Now, the third is equal to the fourth one based on the grounds:
$\beta((\lambda y.yv)z,zv)$.
Thus, the first one is equal to the fourth one based on the grounds:
$\tau(\tau(\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v,(\lambda x.(\lambda y.yx)z)v),\beta((\lambda x.(\lambda y.yx)z)v,(\lambda y.yv)z)),\beta((\lambda y.yv)z,zv)))$.

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

Definition 2.6
The equality theory of Martin Löf's type theory has the following basic proof rules for the Π-type:

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

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

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

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

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

We are finally able to formally define computational paths:

Definition 2.7
Let a and b be elements of a type A. Then, a *computational path* s from a to b is a composition of rewrites (each rewrite is an application of the inference rules of the equality theory of type theory or is a change of bound variables). We denote that by $a=_s b$.

As we have seen in *Example 2.5*, composition of rewrites are applications of the rule τ. Since change of bound variables is possible, each term is considered up to α-equivalence.

3 Identity type

In this section, we have two main objectives. The first one is to propose a formalization to the identity type using computational paths. The second objective is to show how can one use this approach to construct types representing reflexivity, transitivity and symmetry. In the case of the transitive and symmetric types, we also compare our approach with the traditional one, i.e., Martin-Löf's Intensional type. In terms of simplicity, we hope to show the advantage of our approach in the construction of these fundamental types. Since our approach is based on computational paths, we

will sometimes refer to our formulation as the *path-based* approach and the traditional formulation as the *pathless* approach. By this we mean that, even though the Homotopy Type Theory approach to the identity type brings about the notion of paths in the semantics, there is little in the way of handling paths as terms in the language of type theory.

Before the deductions that build the path-based identity type, we would like to make clear that we will use the following construction of the traditional approach [9]:

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

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

3.1 Path-based construction

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

- Formation and Introduction rules:

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

  In our approach, we do not resort to the use of a reflexive constructor r. 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:

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

  We need to clarify the notation being used. First, one should see $h(g)$ as a functional expression h which depends on g. Also, one should notice the the use of ‘´’ in $\acute{g}$. One should see ‘´’ 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 *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 behaviour of *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 $m:Id_A(a,b)$, we can put together all this information in *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 $\acute{g}.h(g)$, resulting in $h(m/g):C$. This behavior is established next by the reduction rule.

- Reduction rule:

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

- Induction rule:

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

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

3.2 Path-based examples

The objective of this subsection is to show how to use in practice the rules that we have just defined. The idea is to show 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 is the fact that reflexive, transitive and symmetric types are essential to the process of building a groupoid model for the identity type [11]. As we shall see, these constructions come naturally from simple computational paths constructed by the application of axioms of the equality of type theory. In contrast, we will also show that constructing the symmetry and transitivity using the operator J can be a little complicated.

Before we start the constructions, we think 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 $g(a,b):Id(a,b)$.

3.2.1 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 $\Pi_{(a:A)}Id_A(a,a)$. The reason is extremely simple: from a term $a:A$,

we obtain the computational path $a =_{\rho} a : A$:

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

3.2.2 Symmetry

The second proposed construction is the symmetry. For the symmetry, we will show two constructions. The first one will be done using the constructor J and the second one using our approach. Our objective is to show the difference of complexity in the process of finding a suitable reason. Both approaches have the objective of constructing a term for the type $\Pi_{(a:A)}\Pi_{(b:A)}(Id_A(a,b) \to Id_A(b,a))$.

Looking at the elimination rule of the constructor J, it is clear that our main objective is to find a suitable $C(x,y,z)$, with $x:A$, $y:A$ and $z:Id_A(a,b)$. The main problem is the fact that, to find the correct reason, we do not have a fixed process or a fixed set of rules to follow. One needs to rely on one's intuition. In this case, one could conclude that the correct reason is to look at $C(x,y,,z)$ as the type $Id_A(y,x)$, with $x:A$, $y:A$ and z as any term (in the deduction, z will be represented by $-$, to indicate that it can be any term). Thus, we obtain the following deduction:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a:A} \quad \dfrac{}{b:A} \quad \dfrac{}{c:Id_A(a,b)} \quad \dfrac{[x:A]}{r(x):Id(x,x)} \quad \dfrac{[x:A, y:A, -:Id_A(a,b)]}{Id_A(y,x)\ \text{type}}\, Id-E}{J(c,r(x)) : Id(b,a)}\, \to -I}{\lambda c.J(c,r(x)) : Id(a,b) \to Id(b,a)}\, \Pi - I}{\lambda b.\lambda c.J(c,r(x)) : \Pi_{(b:A)}(Id_A(a,b) \to Id_A(b,a))}\, \Pi - I}{\lambda a.\lambda b.\lambda c.J(c,r(x)) : \Pi_{(a:A)}\Pi_{(b:A)}(Id_A(a,b) \to Id_A(b,a))}$$

Now, we construct a proof using computational paths. As expected, we need to find a suitable reason. Starting from $a =_t b$, we could look at the axioms of *Definition 2.3* to plan our next step. One of those axioms makes the symmetry clear: the σ axiom. If we apply σ, we will obtain $b =_{\sigma(t)} a$. From this, we can then infer that Id_A is inhabited by $(\sigma(t))(b,a)$. Now, it is just a matter of applying the elimination:

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

3.2.3 Transitivity

The third and last construction will be the transitivity. Both approaches have the objective of constructing a term for the type $\Pi_{(a:A)}\Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a,b)\to Id_A(b,c)\to Id_A(a,c))$.

Let us start with the construction based on J. The proof will be based on the one found in [18]. The difference is that instead of defining induction principles for J based on the elimination rules, we will use the rule directly. The complexity is the same, since the proofs are two forms of presenting the same thing and they share the same reasons. As one should expect, the first and main step is to find a suitable reason. In other words, we need to find suitable $x,y:A$ and $z:Id_A(a,b)$ to construct an adequate $C(x,y,z)$. Similar to the case of symmetry, this first step is already problematic. Different from our approach, in which one starts from a path and applies intuitive equality axioms to find a suitable reason, there is no clear point of how one should proceed to find a suitable reason for the construction based on J. In this case, one should rely on intuition and make attempts until one finds out the correct reason. As one can check in [18], a suitable reason would be $x:A,y:A,-:Id_A(x,y)$ and $C(x,y,z)\equiv Id_A(y,c)\to Id_A(x,c)$. Looking closely, the proof is not over yet. The problem is the type of $C(x,x,r(x))$. With this reason, we have that $C(x,x,r(x))\equiv Id_A(x,c)\to Id_A(x,c)$. Therefore, we cannot assume that $q(x):Id_A(x,c)\to Id_A(x,c)$ is the term $r(x)$. The only way to proceed is to apply again the constructor J to build the term $q(x)$. It means, of course, that we will need to find yet another reason to build this type. This second reason is given by $x:A,y:A,-:Id_A(x,y)$ and $C'(x,y,z)\equiv Id_A(x,y)$. In that case, $C'(x,x,r(x))=Id_A(x,x)$. We will not need to use J again, since now we have that $r(x):Id_A(x,x)$. Then, we can construct $q(x)$:

$$\dfrac{\dfrac{x:A \quad c:A \quad q:Id_A(x,c) \quad \overset{[x:A]}{r(x):Id_A(x,x)} \quad \overset{[x:A,y:A,-:Id_A(x,y)]}{Id_A(x,y)\ \text{type}}}{J(q,r(x)):Id_A(x,c)}\,Id-E}{\lambda q.J(q,r(x)):Id_A(x,c)\to Id_A(x,c)}\to -I \quad .$$

We can finally obtain the desired term:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{a:A \quad b:A \quad p:Id_A(a,b) \quad \overset{[x:A]}{\lambda q.J(q,r(x)):Id_A(x,c)\to Id_A(x,c)} \quad \overset{[x:A,y:A,-:Id_A(x,y)]}{Id_A(y,c)\to Id_A(x,c)\ \text{type}}}{J(p,\lambda q.J(q,r(x))):Id_A(b,c)\to Id_A(a,c)}\,Id-E}{\lambda p.J(p,\lambda q.J(q,r(x))):Id_A(a,b)\to Id_A(b,c)\to Id_A(a,c)}\to -I}{\lambda c.\lambda p.J(p,\lambda q.J(q,r(x))):\Pi_{(c:A)}(Id_A(a,b)\to Id_A(b,c)\to Id_A(a,c))}\,\Pi-I}{\lambda b.\lambda c.\lambda p.J(p,\lambda q.J(q,r(x))):\Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a,b)\to Id_A(b,c)\to Id_A(a,c))}\,\Pi-I}{\lambda a.\lambda b.\lambda c.\lambda p.J(p,\lambda q.J(q,r(x))):\Pi_{(a:A)}\Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a,b)\to Id_A(b,c)\to Id_A(a,c))}\,\Pi-I$$

This construction is an example that makes clear the difficulties of working with the pathless formulation. We had to find two different reasons and use two applications of the elimination rule. Another problem is the fact that the reasons were not obtained by a fixed process, like the applications of axioms in some entity of type theory. They were obtained purely by the intuition that a certain $C(x,y,z)$ should be capable of constructing the desired term. For that reason, obtaining these reasons can be troublesome.

We finish our constructions by giving the path-based construction of the transitivity. 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 $a =_t b$ and $b =_u c$ and then, from these paths, we should conclude that there is a path z that establishes that $a =_z c$. To obtain z, we could try to apply the axioms of *Definition 2.3*. Looking at the axioms, one is exactly what we want: the axiom τ. If we apply τ to $a =_t b$ and $b =_u c$, 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:

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

$$\lambda a.\lambda b.\lambda c.\lambda w.\lambda s.REWR(w(a,b),\acute{t}REWR(s(b,c),\acute{u}(\tau(t,u))(a,c))):\Pi_{(a:A)}\Pi_{(b:A)}\Pi_{(c:A)}(Id_A(a,b)\to Id_A(b,c)\to Id_A(a,c))\;\Pi-I$$

As one can see, each step is just straightforward applications of introduction, elimination rules and abstractions. The only idea behind this construction is just the simple fact that the axiom τ guarantees the transitivity of paths. If one compares the reason of this construction to the one that used J, one can clearly conclude that the reason of the path-based approach was obtained more naturally.

4 A term rewriting system for paths

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

EXAMPLE 4.1
Consider the path $a =_t b : A$. By the symmetric property, we obtain $b =_{\sigma(t)} a : A$. What if we apply the property again on the path $\sigma(t)$? We would obtain a path $a =_{\sigma(\sigma(t))} b : A$. Since we applied symmetry twice in succession, we obtained a path that is equivalent to the initial path t. For that reason, we conclude the act of applying symmetry twice in succession is a redundancy. We say that the path $\sigma(\sigma(t))$ can be reduced to the path t.

As one could see in the aforementioned example, different paths should be considered equal if one is just a redundant form of the other. The example that we have just seen is just a straightforward and simple case. Since the equality theory has a total of 7 axioms, the possibility of combinations that could generate redundancies are high. Fortunately, all possible redundancies were thoroughly mapped by [2]. In this work, a system that establishes all redundancies and creates rules that solve them was proposed. This system, known as $LND_{EQ} - TRS$, maps a total of 39 rules that solve redundancies. To achieve our groupoid structure, we will not need to use all redundancies rules, but only a very specific subset of these rules. One can check the complete system in Appendix B (all rules have been taken from [2, 5]):

- Rules involving σ and ρ

$$\frac{x =_{\rho} x : A}{x =_{\sigma(\rho)} x : A} \quad \rhd_{sr} \quad x =_{\rho} x : A$$

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

- Rules involving τ

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

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

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

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

- Rule involving τ and τ

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

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

DEFINITION 4.2
An *rw*-rule is any of the rules defined in $LND_{EQ} - TRS$.

DEFINITION 4.3
Let s and t be computational paths. We say that $s \rhd_{1rw} t$ (read as: s *rw*-contracts to t) iff we can obtain t from s by an application of only one *rw*-rule. If s can be reduced to t by finite number of *rw*-contractions, then we say that $s \rhd_{rw} t$ (read as s *rw*-reduces to t).

DEFINITION 4.4
Let s and t be computational paths. We say that $s =_{rw} t$ (read as: s is *rw*-equal to t) iff t can be obtained from s by a finite (perhaps empty) series of *rw*-contractions and reversed *rw*-contractions. In other words, $s =_{rw} t$ iff there exists a sequence $R_0, ..., R_n$, with $n \geq 0$, such that

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

PROPOSITION 4.5
Rewrite equality is transitive, symmetric and reflexive.

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

Before talking about higher $LND_{EQ} - TRS$ systems, we would like to mention that $LND_{EQ} - TRS$ is terminating and confluent. The proof of this affirmation can be found in [2–4, 8].

4.1 $LND_{EQ} - TRS_2$

Until now, this section has concluded that there exist redundancies which are resolved by a system called $LND_{EQ} - TRS$. This system establishes rules that reduces these redundancies. Moreover, we concluded that these redundancies are just redundant uses of the equality axioms showed in *Section 2*. In fact, since these axioms just define an equality theory for type theory, one can specify and say that these are redundancies of the equality of type theory. As we mentioned, the $LND_{EQ} - TRS$ has a total of 39 rules [2]. Since the *rw*-equality is based on the rules of $LND_{EQ} - TRS$, one can just imagine the high number of redundancies that *rw*-equality could cause. In fact, a thoroughly study of all the redundancies caused by these rules could generate an entire new work. Fortunately, we are only interested in the redundancies caused by the fact that *rw*-equality is transitive, reflexive and symmetric with the addition of only one specific rw_2-rule. Let us say that we have a system, called $LND_{EQ} - TRS_2$, that resolves all the redundancies caused by *rw*-equality (the same way that $LND_{EQ} - TRS$ 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 these properties. Since *rw*-equality is just a sequence of *rw*-rules (also similar to equality, since equality is just a computational path, i.e., a sequence of identifiers), then we could put a name on these sequences. For example, if s and t are *rw*-equal because there exists a sequence $\theta : R_0, ..., R_n$ that justifies the *rw*-equality, then we can write that $s =_{rw_\theta} t$. Thus, we can rewrite, using *rw*-equality, all the rules that originated the rules involving τ, σ and ρ. For example, we have:

$$\dfrac{\dfrac{x =_{rw_t} y : A \qquad y =_{rw_r} w : A}{x =_{rw_{\tau(t,r)}} w : A} \qquad w =_{rw_s} z : A}{x =_{rw_{\tau(\tau(t,r),s)}} z : A}$$

$$\rhd_{tt_2} \dfrac{x =_{rw_t} y : A \qquad \dfrac{y =_{rw_r} w : A \qquad w =_{rw_s} z : A}{y =_{rw_{\tau(r,s)}} z : A}}{x =_{rw_{\tau(t,\tau(r,s))}} z : A}$$

Therefore, we obtain the rule tt_2, that 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 page 5. In other words, we have tr_2, tsr_2, trr_2, tlr_2, sr_2, ss_2 and tt_2. Since we have now 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 a 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 is two possible *rw*-sequences that reduces this case: the first one is $\theta:\tau(s,t)\rhd_{1rw}\tau(s',t)\rhd_{1rw}\tau(s',t')$ and the second $\theta':\tau(s,t)\rhd_{1rw}\tau(s,t')\rhd_{1rw}\tau(s',t')$. 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 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 4.6
Let θ and ϕ be *rw*-equalities expressed by two *rw*-sequences: $\theta:\theta_1,...,\theta_n$, 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 $\tau(\theta_n,\theta_m)$ described by the following process: $t\in T$ is of the form $\tau(\theta_{l_1},\phi_{r_1})\rhd_{1rw}\tau(\theta_{l_2},\phi_{r_2})\rhd_{1rw}...\rhd_{1rw}\tau(\theta_{l_x},\phi_{r_y})$, 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 $l_{i+1}=l_i$ and $r_{i+1}=1+r_i$. 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 $y=_{cd_2}x$.

Proposition 4.7
rw_2-equality is transitive, symmetric and reflexive.

Proof. Analogous to Proposition 4.5. ∎

4.2 Globular structure

In the previous subsection, we have just showen the existence of paths between computational paths. Such paths were originated from redundancies caused by *rw*-equality. Analogously, one could think of redundancies caused by rw_2-equality, which would be resolved by the establishment of computational paths between two equivalent rw_2-equalities. With that in mind, we can extend this process up to the infinity. That way, we obtain an infinite number of rw_n-equalities and $LND_{EQ}-TRS_n$ systems. There is a mathematical entity suitable to express this fact. It is known as *globular set*.

Definition 4.8
The *level* of a path is defined as follows:

- If $a,b:A$ are terms such that a and b are not *rw*-equalities, then we say that a path $a=_s b$ has level 1.
- If $r,s:A$ are rw_n-equalities, then a path $r=_\theta s$ has level $n+1$.

The idea is that if we have two n-level paths r and s and if r can be rewritten into s by a sequence of rewrites established by $LND_{EQ}-TRS_n$, then this sequence of rewrites is a $(n+1)$-level path that establishes $r=_{rw_n}s$. Consider now the following structure [15]:

DEFINITION 4.9
Let $n \in \mathbb{N}$. An n-globular set X is the following diagram of sets and functions:

$$X(n) \underset{t}{\overset{s}{\rightrightarrows}} X(n-1) \underset{t}{\overset{s}{\rightrightarrows}} \cdots \underset{t}{\overset{s}{\rightrightarrows}} X(0).$$

The diagram has to obey the following equations for all $x \in \{2, ..., n\}$ and $x \in X(m)$:

$$s(s(x)) = s(t(x)), \quad t(s(x)) = t(t(x)).$$

Computational paths can form a globular-set structure. To see this, consider $X(n)$ as the set of n-level paths between two $(n-1)$-level paths (for $n = 0$, just consider two non-path objects) and for any path $x =_m y$, consider that $s(m) = x$ and $t(m) = y$. That way, if $n \geq 2$ and $m \in X(n)$, then $s(m) \in X(n-1)$ and $t(m) \in X(n-1)$. Now, if we consider two n-level paths $t =_\theta m$, and $t =_\alpha m$ and a $(n+1)$-level path $\theta =_\phi \alpha$, the following equations hold:

$$s(s(\phi)) = s(\theta) = t = s(\alpha) = s(t(\phi)).$$
$$t(s(\phi)) = t(\theta) = m = t(\alpha) = t(t(\phi)).$$

All globular-set conditions hold. Then, we can think that computational paths form a globular set structure all up to infinity, i.e. a ∞-globular-set.

5 The groupoid induced by computational paths

The relation between the algebraic structure of groupoid and the identity type was first noticed by [11]. In this work, the authors claim that it is possible to think of any type of type theory as having a groupoid structure. To do that, one could think of the terms of a type as objects and the propositional identities between these terms to be morphisms. Using J, [11] showed that the following types, which correspond to the groupoid equations, are inhabited:

- $Id_{Id_A(x,z)}(trans(trans(p,q),r), trans(p, trans(q,r)))$
- $Id_{Id_A(x,y)}(trans(refl(x),r),r)$
- $Id_{Id_A(x,y)}(trans(r,refl(x)),r)$
- $Id_{Id_A(y,y)}(trans(symm(r),r),refl(x))$
- $Id_{Id_A(x,x)}(trans(r,symm(r)),refl(x))$
- $Id_{Id_A(x,y)}(symm(symm(r)),r)$.

After this important work, the authors further develop their ideas in [12], showing how J can be interpreted categorically.

Since our approach is not based on J, but on the fact that identities are established by computational paths, our objective in this section is to show that computational paths, together with the reduction rules discussed in the last section, are also capable of inducing groupoidal structures.

5.1 The groupoid of computational paths

Before we conclude that computational paths induces categories with groupoid properties, we need to make clear the difference between a strict and a weak category. As one will see, the word weak will appear many times. This will be the case because some of the categorical equalities will not hold 'on the nose', so to say. They will hold up to *rw*-equality or up to higher levels of *rw*-equalities.

This is similar to the groupoid model of the identity type proposed by [11]. In [11], the equalities do not hold 'on the nose', they hold up to propositional equality (or up to homotopy if one uses the homotopy interpretation). To indicate that these equalities hold only up to some property, we say that the induced structure is a weak categorical structure. This will be even more clear in the proof of the following proposition:

PROPOSITION 5.1
For each type A, computational paths induces a weak categorical structure called A_{rw}.

PROOF. Since we are working with categories, a good understanding of the basic concept of what is a category is essential. If one is not familiarized with the concept, one should check [1]. To sum up, a category is a structure with objects and morphisms between these objects. These morphisms must obey two main laws: the associative and identity laws. To define a weak categorical structure A_{rw}, the first step is to define the objects, the morphisms:

- Objects: the objects of A_{rw} are terms a of the type A, i.e., $a:A$.
- Morphisms: a morphism (arrow) between terms $a:A$ and $b:A$ are arrows $s:a\to b$ such that s is a computational path between the terms, i.e., $a=_s b:A$.

We need now to define the composition of morphisms, the identity morphism and check the associative and identity laws. Since we are working in a weak context, the associative and identity law needs only to hold up to *rw*-equality.

- Compositions: given arrows (paths) $s:a\to b$ and $r:b\to c$, we need to find an arrow $t:a\to c$ such that $t=r\circ s$. To do that, we first need to define the meaning of a composition of paths. Since equality has the transitive axiom, it is natural to define the composition as an application of the transitivity, i.e. $t=r\circ s=\tau(s,r)$. Therefore, for any $s:a\to b$ and $r:b\to c$, we always have a $t=r\circ s=\tau(s,r)$.
- Associativity of the composition: given arrows $s:a\to b$, $r:b\to c$ and $t:c\to d$, since we are working with a weak categorical structure, we need to conclude that $t\circ(r\circ s)=_{rw}(t\circ r)\circ s$. Substituting the compositions by the corresponding transitivities, we need to conclude that $\tau(\tau(s,r),t)=_{rw}\tau(s,\tau(r,t))$. This is a direct consequence of the rule tt: $\tau(\tau(s,r),t)\rhd_{tt}\tau(s,\tau(r,t))$. Therefore, we conclude that $\tau(\tau(s,r),t)=_{rw}\tau(s,\tau(r,t))$.
- Identity morphism: for any object a, consider the reflexive path $a=_\rho a$ as the identity arrow 1_a. Let us call this path as ρ_a (to indicate that it is a reflexive path of a).
- Identity law: for any arrows (paths) $s:a\to b$, we need to show that $s\circ 1_a=s=1_b\circ s$. This can be shown (also in a weak sense) with a straightforward application of *rw*-rules. We have that $s\circ 1_a=s\circ\rho_a=\tau(\rho_a,s)\rhd_{tlr}s$, therefore $\tau(\rho_a,s)=_{rw}s$. For the other equation, we have that $1_b\circ s=\rho_b\circ s=\tau(s,\rho_b)\rhd_{trr}s$ and thus, $\tau(s,\rho_b)=_{rw}s$.

Since all these conditions have been satisfied, we conclude that the structure A_{rw} induced by computational paths is a weak categorical structure. ∎

Is it possible to induce a strict (i.e. one that the equalities will hold 'on the nose') using computational paths? The answer is *yes*. Since *rw*-equality is transitive, symmetric and reflexive, *rw*-equality is an equivalence relation. If we use equivalent classes of *rw*-equalities as arrows, the equalities will hold 'on the nose'. Since we are trying to prove similar results to the ones obtained for identity types by [11] (in [11], as mentioned before, the equalities only in a weak sense), we are more interested in this weak categorical structure that we have just obtained. The definition of an arrow isomorphism follows [1]:

DEFINITION 5.2
Let $f:A\to B$ be an arrow of any category. f is called an isomorphism if there exists a $g:B\to A$ such that $g\circ f=1_A$ and $f\circ g=1_B$. g is called the inverse of f and can be written as f^{-1}.

We can now finally define a categorical groupoid [1]:

DEFINITION 5.3
A groupoid is a category in which every arrow is an isomorphism.

PROPOSITION 5.4
The induced structure A_{rw} has a weak groupoidal structure.

PROOF. We need to show that every arrow is an isomorphism. Since we are working in a weak sense, the isomorphism equalities need only to hold up to *rw*-equality. To show that, for every arrow $s:a\to b$, we need to show a $t:b\to a$ such that $t\circ s=_{rw}1_a$ and $s\circ t=_{rw}1_b$. To do that, recall that every computational path has an inverse path $\sigma(s)$. Put $t=\sigma(s)$. Thus, we have that $s\circ t=s\circ\sigma(s)=\tau(\sigma(s),s)\rhd_{tsr}\rho_b$. Since $\rho_b=1_b$, we conclude that $s\circ t=_{rw}1_b$. We also have that $t\circ s=\sigma(s)\circ s=\tau(s,\sigma(s))\rhd_{tr}\rho_a$. Therefore, $t\circ s=1_a$. We conclude that every arrow is a weak isomorphism and thus, A_{rw} is a weak groupoidal structure. ∎

If we work with classes of equivalences as arrows, we can obtain a groupoidal structure in the strict sense. With that, we finally conclude that computational paths have a groupoid model. As one could notice, we only needed to use the properties of computational paths, together with rules of the rewrites $LND_{EQ}-TRS$. Having these concepts and rules, we have just needed to show that computational paths induces simple structures of category theory. The groupoidal model came naturally from the concept of computational paths, without the need of constructing any complex term of an identity type, as done in [11]. The fact that computational paths have a natural groupoidal model is an advantage, since it will automatically imply that the identity type has a natural groupoidal model (without needing to build any term using a constructor, like the constructor J of the traditional identity type).

5.2 Higher structures

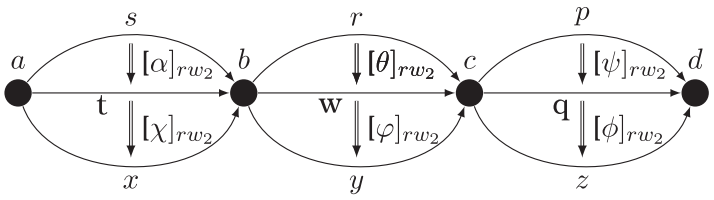
We have just shown that computational paths induce a weak groupoidal structure known as A_{rw}. We also know that the arrows (or morphisms) of A_{rw} are computational paths between two terms of a type A. As we saw in the previous section, sometimes a computational path can be reduced to another by rules that we called *rw*-rules. That way, if we have terms $a,b:A$ and paths between these terms, we can define a new structure. This new structure, called $A_{2rw}(a,b)$, has objects as computational paths between a and b and there is a morphism between paths $a=_s b$ and $a=_t b$ iff $s=_{rw}t$. Since *rw*-equality is transitive, reflexive and symmetric, A_{2rw} is a weak categorical structure which the equalities hold up to rw_2-equality. The proof of this fact is analogous to the *Proposition 4.1*. The sole difference is the fact that since the morphisms are *rw*-equalities, instead of computational paths, all the equalities will hold up to rw_2-equality. To see this, take the example of the associativity. Looking at the $LND_{EQ}-TRS_2$ system, we have that $\tau(\tau(\theta,\sigma),\phi)\rhd_{tt_2}\tau(\theta,\tau(\sigma,\phi))$ (θ, σ and ϕ represent *rw*-equalities between paths from a to b). Therefore, $\tau(\tau(\theta,\sigma),\phi)=_{rw_2}\tau(\theta,\tau(\sigma,\phi))$. The associative law holds up to rw_2-equality. As one can easily check, the identity law will also hold up to rw_2-equality. Therefore, $A_{2rw}(a,b)$ has a weak categorical structure. Analogous to *Proposition 4.4*, the groupoid law will also hold up to rw_2-equality.

Instead of considering A_{rw} and $A_{2rw}(a,b)$ as separated structures, we can think of a unique structure with 2-levels. The first level is A_{rw} and the second one is the structure $A_{2rw}(a,b)$ between each pair

of objects $a, b \in A_{rw}$. We call this structure $2-A_{rw}$. The morphisms of the first level are called 1-morphisms and the ones of the second level are called 2-morphisms (also known as 2-arrows or 2-cells). Since it has multiple levels, it is considered a higher structure. We want to prove that this structure is a categorical structure known as weak 2-category. The main problem is the fact that in a weak 2-category, the last level (i.e. the second level) needs to hold up in a strict sense. This is not the case for $2-A_{2rw}$, since each $A_{2rw}(a,b)$ only holds up to rw_2-equality. Nevertheless, there still a way to induce this weak 2-category. Since rw-equality is an equivalence relation (because it is transitive, symmetric and reflexive), we can consider a special $A_{2rw}(a,b)$, where the arrows are the arrows of $A_{2rw}(a,b)$ modulo rw_2-equality. That way, since the equalities hold up to rw_2-equality in $A_{2rw}(a,b)$, they will hold in a strict sense when we consider the equivalence classes of rw_2-equality. We call this structure $[A_{2rw}(a,b)]$. In this structure, consider the composition of arrows defined as: $[\theta]_{rw_2} \circ [\phi]_{rw_2} = [\theta \circ \phi]_{rw_2}$. Now, we can think of the structure $[2-A_{rw}]$. This structure is similar to $2-A_{rw}$. The difference is that the categories of the second level are $[A_{2rw}(a,b)]$ instead of $A_{2rw}(a,b)$. We can now prove that $[2-A_{rw}]$ is a weak 2-category:

PROPOSITION 5.5
Computational paths induce a weak 2-category called $[2-A_{rw}]$.

PROOF. First of all, let us draw a diagram that represents $[2-A_{rw}]$:



In this diagram, we represent 1-arrows and 2-arrows between these 1-arrows. The fact that 2-arrows are equivalence classes is represented by the brackets.

Since we are working with a higher structures, some new properties should be checked. One of these properties is the fact that 2-arrows can be composed horizontally [14]. In other words, given 1-morphisms $s: a \to b$, $r: b \to c$, $t: a \to b$, $w: b \to c$ and 2-morphisms $[\alpha]_{rw_2}: s \to t$ and $[\theta]_{rw_2}: r \to w$, one should be able to define a horizontal composition $\circ_h$: $([\theta]_{rw_2} \circ_h [\alpha]_{rw_2}): r \circ s \to w \circ t$.

Given $[\alpha]_{rw_2}: [s = \alpha_1, ..., \alpha_n = t]$ and $[\theta]_{rw_2}: [r = \theta_1, ..., \theta_m = w]$, then we define the horizontal composition $([\theta]_{rw_2} \circ_h [\alpha]_{rw_2})$ as the sequence $[\tau(s = \alpha_1, r = \theta_1), ..., \tau(\alpha_n, \theta_1) ..., \tau(\alpha_n = t, \theta_m = w)]_{rw_2}$.

We also need to verify the associative and identity law for $\circ_h$. Since we are working with a weak 2-category, these laws should hold up to natural isomorphism [14]. To verify these laws, the idea is that every 2-morphism of $[A_{rw_2}]$ is an isomorphism. To see that, remember that rw-equality is transitive, symmetric and reflexive. Therefore, if we have $[\theta]_{rw_2}$, we can think of the inverse $[\sigma(\theta)]_{rw_2}$. If we compose them, we have that $[\theta]_{rw_2} \circ [\sigma(\theta)]_{rw_2} = [\theta \circ \sigma(\theta)]_{rw_2}$. Since we have in $LND_{EQ} - TRS_2$ that $(\theta \circ \sigma(\theta)) = \tau(\sigma(\theta), \theta)) \rhd_{tsr_2} \rho_r$, then $\theta \circ \sigma(\theta) =_{rw_2} \rho_r$ and thus, $[\theta \circ \sigma(\theta)]_{rw_2} = [\rho_r]_{rw_2}$. Analogously, one can prove that $[\sigma(\theta) \circ \theta]_{rw_2} = [\rho_w]_{rw_2}$. Therefore, every 2-morphism of $[A_{rw_2}]$ is an isomorphism. Since a natural transformation is a natural isomorphism iff every component is an isomorphism (as one can check in [1]), we conclude that finding isomorphisms for the associative and identity laws is just a matter of finding the correct morphisms.

For the associative law, we need to check that there is a natural isomorphism between $(([\psi]_{rw_2} \circ_h [\theta]_{rw_2}) \circ_h [\alpha]_{rw_2})$ and $([\psi]_{rw_2} \circ_h ([\theta]_{rw_2} \circ_h [\alpha]_{rw_2}))$. To do this, by the definition of horizontal composition, a component of $(([\psi]_{rw_2} \circ_h [\theta]_{rw_2})$ is a term of the form $\tau(\alpha_x, \tau(\theta_y, \psi_z))$, with x, y, and z being suitable natural numbers that respect the order of the horizontal composition. Analogously, the same

component of $([\psi]_{rw_2} \circ_h ([\theta]_{rw_2} \circ_h [\alpha]_{rw_2}))$ is just a suitable term $\tau(\tau(\alpha_x, \theta_y), \psi_z)$. The isomorphism between these component is clearly established by the inverse *tt* rule, i.e. $\tau(\alpha_x, \tau(\theta_y, \psi_z)) =_{rw_{\sigma(tt)}} \tau(\tau(\alpha_x, \theta_y), \psi_z)$

The identity laws use the same idea. We need to check that $([\alpha]_{rw_2} \circ_h [\rho_{\rho_a}]_{rw_2}) = [\alpha]_{rw_2}$. To do that, we need to take components $(\rho_{\rho_a}, \alpha_y)$ and α_y and establish their isomorphism: $(\rho_{\rho_a}, \alpha_y) =_{rw_{tlr}} \alpha_y$.

The other natural isomorphism, i.e. the isomorphism between $([\rho_{\rho_b}]_{rw_2} \circ_h [\alpha]_{rw_2})$ and $[\alpha]_{rw_2}$ can be established in an analogous way, just using the rule *trr* instead of *tlr*. Just for purpose of clarification, ρ_{ρ_a} comes from the reflexive property of *rw*-equality. Since ρ_a is the identity path, using the reflexivity we establish that $\rho_a =_{rw} \rho_a$, generating ρ_{ρ_a}.

We call the associative morphism generated by $\sigma(tt)$ as *assoc*, the morphism generate by *tlr* as l_s^* and the one generated by *trr* as r_s^*. With the associative and identity isomorphisms established, we now need to check the *interchange law* [14]. We need to check that:

$$([\varphi]_{rw_2} \circ [\theta]_{rw_2}) \circ_h ([\chi]_{rw_2} \circ [\alpha]_{rw_2}) = ([\varphi]_{rw_2} \circ_h [\chi]_{rw_2}) \circ ([\theta]_{rw_2} \circ_h [\alpha]_{rw_2}).$$

From $(([\varphi]_{rw_2} \circ [\theta]_{rw_2}) \circ_h ([\chi]_{rw_2} \circ [\alpha]_{rw_2}))$, we have:

$$\begin{gathered}(([\varphi]_{rw_2} \circ [\theta]_{rw_2}) \circ_h ([\chi]_{rw_2} \circ [\alpha]_{rw_2})) = \\ [\tau(\theta, \varphi)]_{rw_2} \circ_h [\tau(\alpha, \chi)]_{rw_2} = \\ [\theta_1, \ldots, \theta_n = \varphi_1, \ldots, \varphi_{n'}]_{rw_2} \circ_h [(\alpha_1, \ldots, \alpha_m = \chi_1, \ldots \chi_{m'}]_{rw_2} = \\ [\tau(\alpha_1, \theta_1), \ldots, \tau(\alpha_m = \chi_1, \theta_1), \ldots, \tau(\chi_n, \theta_1), \ldots, \tau(\chi_n, \theta_{m'} = \varphi_1), \ldots, \tau(\chi_n, \varphi_{n'})]_{rw_2}.\end{gathered}$$

From $(([\varphi]_{rw_2} \circ_h [\chi]_{rw_2}) \circ ([\theta]_{rw_2} \circ_h [\alpha]_{rw_2}))$:

$$\begin{gathered}(([\varphi]_{rw_2} \circ_h [\chi]_{rw_2}) \circ ([\theta]_{rw_2} \circ_h [\alpha]_{rw_2}))(r \circ s) = \\ ([\tau(\chi_1, \varphi_1), \ldots, \tau(\chi_n, \varphi_1), \ldots, \tau(\chi_n, \varphi_{n'})]_{rw_2} \circ [\tau(\alpha_1, \theta_1), \ldots, \tau(\alpha_m, \theta_1), \ldots, \tau(\alpha_m, \theta_{m'})]_{rw_2} = \\ [\tau(\alpha_1, \theta_1), \ldots, \tau(\alpha_m, \theta_1), \ldots, \tau(\alpha_m, \theta_{m'}), \ldots, \tau(\chi_1, \varphi_1), \ldots, \tau(\chi_n, \varphi_1), \ldots, \tau(\chi_n, \varphi_{n'})]_{rw_2}.\end{gathered}$$

If one looks closely, one can notice that this is a suitable to apply cd_2. Individually, every variable that appears in the sequence of transitivities follows the same expansion in both cases. The only difference is how the choices have been made. That way, if we construct T, as defined in Definition 4.6, one can check that $\tau(\alpha_1, \theta_1), \ldots, \tau(\alpha_m = \chi_1, \theta_1), \ldots, \tau(\chi_n, \theta_1), \ldots, \tau(\chi_n, \theta_{m'} = \varphi_1), \ldots, \tau(\chi_n, \varphi_{n'}) \in T$ and $\tau(\alpha_1, \theta_1), \ldots, \tau(\alpha_m, \theta_1), \ldots, \tau(\alpha_m, \theta_{m'}), \ldots, \tau(\chi_1, \varphi_1), \ldots, \tau(\chi_n, \varphi_1), \ldots, \tau(\chi_n, \varphi_{n'}) \in T$ For that reason, we establish:

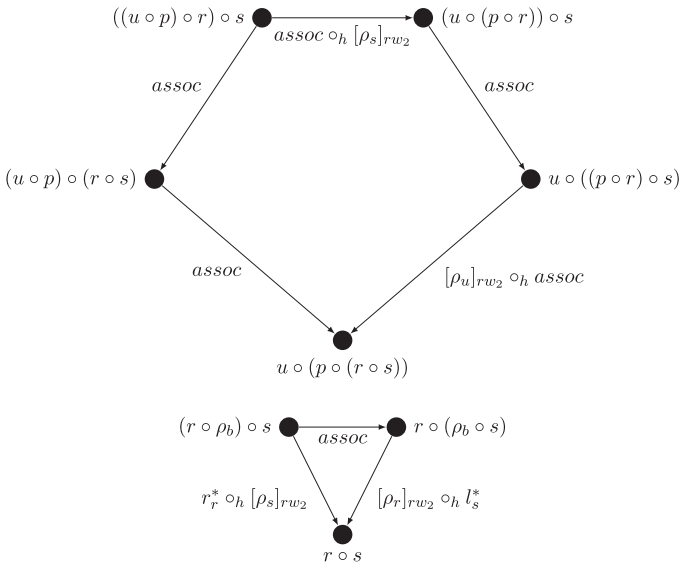
$$\begin{gathered}[\tau(\alpha_1, \theta_1), \ldots, \tau(\alpha_m = \chi_1, \theta_1), \ldots, \tau(\chi_n, \theta_1), \ldots, \tau(\chi_n, \theta_{m'} = \varphi_1), \ldots, \tau(\chi_n, \varphi_{n'})]_{rw_2} =_{cd_2} \\ [\tau(\alpha_1, \theta_1), \ldots, \tau(\alpha_m, \theta_1), \ldots, \tau(\alpha_m, \theta_{m'}), \ldots, \tau(\chi_1, \varphi_1), \ldots, \tau(\chi_n, \varphi_1), \ldots, \tau(\chi_n, \varphi_{n'})]_{rw_2}.\end{gathered}$$

Since we are working with equivalence classes, this equality holds strictly.

To end the proof, there is one more thing that we should verify. Since we are working with a weak structure, there is some laws that should hold. These laws are known as *coherence laws*. In a weak 2-category, the coherence laws are the following fact [14]:

Given the following diagram of $1-$*morphisms*:

The following diagrams should commute:



The proofs are straightforward. Remember that *assoc* is just an application of $\sigma(tt)$, r_r^* is an application of *trr* and l_s^* an application of *tlr*. First, let us start with $((u\circ p)\circ r)\circ s = \tau(s,\tau(r,\tau(p,u)))$ going to the right of the diagram:

$$(assoc\circ_h[\rho_s]_{rw_2})(\tau(s,\tau(r,\tau(p,u)))) = \tau(s,assoc(\tau(r,\tau(p,u)))) = \tau(s,\tau(\tau(r,p),u))$$
$$assoc(\tau(s,\tau(\tau(r,p),u))) = \tau(\tau(s,\tau(r,p)),u)$$
$$([\rho_u]_{rw_2}\circ_h assoc)(\tau(\tau(s,\tau(r,p)),u)) = \tau(assoc(\tau(s,\tau(r,p))),u) = \tau(\tau(\tau(s,r),p)),u) = u\circ(p\circ(r\circ s)).$$

Now, starting from the same $\tau(s,\tau(r,\tau(p,u)))$ and going bottom left:

$$assoc(\tau(s,\tau(r,\tau(p,u)))) = \tau(\tau(s,r),\tau(p,u))$$
$$assoc(\tau(\tau(s,r),\tau(p,u))) = \tau(\tau(\tau(s,r),p),u) = u\circ(p\circ(r\circ s)).$$

Therefore, the first diagram commutes. We now need to check the second one. Let us start from $((r\circ\rho_b)\circ s) = \tau(s,\tau(\rho_b,r))$ and going to the right of the diagram:

$$assoc(\tau(s,\tau(\rho_b,r))) = \tau(\tau(s,\rho_b),r)$$
$$([\rho_r]_{rw_2}\circ_h l_s^*)\tau(\tau(s,\rho_b),r) = \tau(l_s^*(\tau(s,\rho_b)),r) = \tau(s,r) = r\circ s.$$

Now, starting from the same $\tau(s,\tau(\rho_b,r))$ and going right bottom:

$$(r_r^*\circ_h[\rho_s]_{rw_2})\tau(s,\tau(\rho_b,r)) = \tau(s,r_r^*(\tau(\rho_b,r))) = \tau(s,r) = r\circ s.$$

Thus, the second diagram commutes. The coherence laws hold. We finally finish the proof that $[2-A_{rw}]$ is a weak 2-category. ∎

We can also conclude that $[2-A_{rw}]$ has a weak 2-groupoid structure. That is the case because we already know (from *Proposition 4.4*) that the groupoid laws are satisfied by 1-morphisms up to the

isomorphism of the next level, i.e., up to *rw*-equality and the 2-morphisms, as we have just seen, are isomorphisms (that hold in a strict way, since the second level is using classes of equivalence). With that, we showed that computational paths induces a 2-weak groupoid. If one compares this groupoid with the one obtained by the homotopy interpretation, this 2-weak category is similar to the fundamental weak 2-groupoid of a space S, denoted by $\Pi_2 S$ (that $\Pi_2 S$ forms a weak 2-category can be seen in [15]).

Since we could induce a weak 2-categorical structure using computational paths, would it be possible to induce even higher structures? The answer is *yes*, since we have infinite levels of *rw*-rules established by infinite $LND_{EQ} - TRS_n$ systems. For example, we could add a new level $A_{3rw}(\theta, \alpha)$. where θ and α are 2-morphisms. We would have a structure with 3 levels and we could try to prove that this structure is a weak 3-category. The problem is, as one could see in the proof of *Proposition 4.5*, working with higher structures can be difficult. A weak 3-weak category has more types of compositions than the 2-weak one, since we have an additional level. Moreover, since it is weak, there are coherence laws that must be checked. For a 3-weak category, these laws are much more complicated than the ones for 2-weak categories. For that reason, we will leave the study of higher structures induced by computational paths with more than 2 levels for the future, since it is still work in progress. In fact, our aim is to prove, in a future work, that computational path induces a weak category with infinite levels, known as weak ω-category. In fact, we want to show that this weak infinite category forms a weak ω-groupoid. We believe that it is possible to achieve these results, since it was proved by [16, 19] that the identity type induces such structure. Given the connection between computational paths and terms of identity types, we should be able to prove that computational paths also induces a weak ω-groupoid.

6 Uniqueness of identity proofs

One of the main results that arises from the groupoid interpretation of [12] is that it refutes the uniqueness of identity proofs, also known as *UIP*. [12] defines UIP as the following property:

Definition 6.1
Let a and b be objects of a type A. *UIP* is the following property: given any proofs p and q of proposition a equals to b, then there is always another proof establishing the equality of p and q.

In terms of computational paths, *UIP* asks if, for every pair of objects t and s between a and b of a type A, there is an *rw*-equality $t =_{rw_\theta} s$. To better see this situation, a concrete example is given by [5]:

Example 6.2
Consider the term $(\lambda x.(\lambda y.yx)(\lambda w.zw))v$. We want to reduce this term to the term zv. To achieve this goal, we can follow 3 different paths:

- $(\lambda x.(\lambda y.yx)(\lambda w.zw))v \rhd_\eta (\lambda x.(\lambda y.yx)z)v \rhd_\beta (\lambda y.yv)z \rhd_\beta zv$.
- $(\lambda x.(\lambda y.yx)(\lambda w.zw))v \rhd_\beta (\lambda x.(\lambda w.zw)x)v \rhd_\eta (\lambda x.zx)v \rhd_\beta zv$.
- $(\lambda x.(\lambda y.yx)(\lambda w.zw))v \rhd_\beta (\lambda x.(\lambda w.zw)x)v \rhd_\beta (\lambda w.zw)v \rhd_\eta zv$.

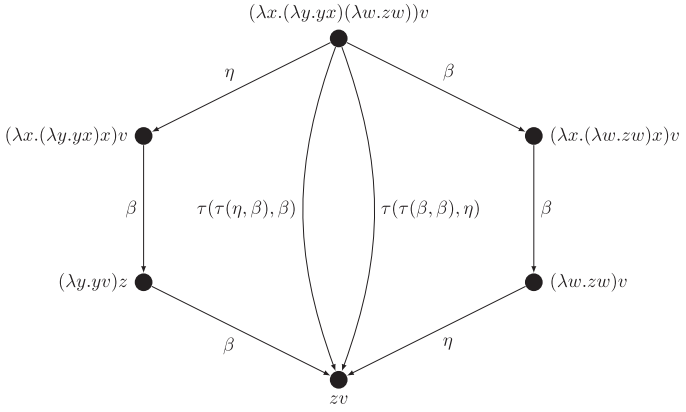
Thus, if *UIP* holds, those 3 paths are just different ways of expressing the same thing. In other words, it would be possible to find *rw*-equalities establishing equalities between them.

The objective of this section is to show that, similarly to the traditional approach for the identity type, our path-based one also refutes *UIP*. Given a type A, by the definition of *UIP*, one can conceive a type $UIP(A)$ that is inhabited iff for any terms $a, b : A$ and paths $a =_s b$ and $a =_t b$, then $s =_{rw} t$.

As stated by [12], if *UIP* holds, then every type A is a trivial groupoid structure in which there is only one morphism between every pair of objects. To show that *UIP* is false, one only needs to construct an A such that there is more than one morphism between a pair of objects (i.e. there is at least one pair of objects $a, b : A$ and a pair of paths s and t between them such that *sthat* $s \neq_{rw} t$). To achieve that, we can take advantage of our previous example, and consider $(\lambda x.(\lambda y.yx)(\lambda w.zw))v : A$. Therefore:

THEOREM 6.3
The type *UIP* is empty.

PROOF. We construct A_{rw}:



As depicted above, there are (at least) two possible paths between $(\lambda x.(\lambda y.yx)(\lambda w.zw))v$ and zv. The first path is given by $\tau(\tau(\eta, \beta), \beta)$ and the second by $\tau(\tau(\beta, \beta), \eta)$. Moreover, looking at all *rw*-rules (check Appendix B), there is no rule that establishes the *rw*-equality between these two paths. That way, A_{rw} is not a trivial groupoid and thus, *UIP* is empty. ∎

With that, we conclude that our approach based on computational paths refutes the uniqueness of identity proofs.

7 Conclusion

Inspired by a recent discovery that the propositional equality between terms can be interpreted as the type of homotopy paths, we have revisited the formulation of the intensional identity type, proposing a approach based on an entity known as *computational path*. We have proposed that a computational path $a =_s b : A$ gives grounds to building a term $s(a, b)$ of the identity type, i.e. $s(a, b) : Id_A(a, b)$, and is formed by a composition of basic rewrites, each with their identifiers taken as constants. We have also developed our approach, showing how the path-based identity type can be rather straightforwardly used in deductions. In particular, we have shown the simplicity of our elimination rule, demonstrating that it is based on path constructions, which are built from applications of simple axioms of the equality for type theory. To make our point even clearer, we have exposed three path-based constructions. More specifically, constructions that prove the transitivity, reflexivity and symmetry of the propositional equality. We have also argued that, for these constructions, the process of finding the reason that allows for building the desired term

was simple and straightforward in our approach. At the same time, in the traditional (or pathless) approach, this is not entirely true, since finding the correct reason was a cumbersome process.

After establishing the foundations of our approach, we analysed one important structure that the traditional identity type induces: the algebraic structure known as groupoid. Our objective was to show that our approach is on a par with the pathless one, i.e. our path-based identity also induces a groupoid structure. To prove that, we have shown that the axioms of equality generate redundancies, which are resolved by paths between paths. We mentioned that there already exists a system called $LND_{EQ} - TRS$ that maps and resolves these redundancies. We have gone further, proposing the existence of a higher $LND_{EQ} - TRS_2$ which resolves redundancies generated by the $LND_{EQ} - TRS$ system. Using $LND_{EQ} - TRS$, we have proved that a computational path is capable of inducing a weak groupoid structure. Using the higher rewriting system, we have induced a structure known as weak 2-groupoid. With that, we believe we have opened the way, in a future work, for a possible proof establishing that computational paths induces a weak ω-groupoid.

We have also shown that, using our groupoid of computational paths, it is possible to refute the principle of uniqueness of identity proofs for the path-based approach. This result ties in with the one obtained by [12] for the traditional identity type.

References

[1] S. Awodey. *Category Theory*. Oxford University Press, 2010.

[2] A. G. de Oliveira. *Proof Transformations for Labelled Natural Deduction via Term Rewriting*'. Master's Thesis, Depto. de Informática, Universidade Federal de Pernambuco, Recife, Brazil, April 1995.

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

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

[5] R. J. G. B. de Queiroz and A. G. de Oliveira. Propositional equality, identity types, and direct computational paths. 2013. http://arxiv.org/abs/1107.1901.

[6] R. J. G. B. de Queiroz and A. G. de Oliveira. Natural deduction for equality: The missing entity, In *Advances in Natural Deduction*, pp. 63–91, Springer, 2014.

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

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

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

[10] J. R. Hindley and J. P. Seldin. *Lambda-Calculus and Combinators: An Introduction*. Cambridge University Press, 2008.

[11] M. Hofmann and T. Streicher. The groupoid model refutes uniqueness of identity proofs. *In Logic in Computer Science, 1994. LICS'94. Proceedings, Symposium on*, pp. 208–212. IEEE, 1994.

[12] M. Hofmann and T. Streicher, The groupoid interpretation of type theory. In *Twenty-five Years of Constructive Type Theory (Venice, 1995)*, Vol. 36 of Oxford Logic Guides, pp. 83–111. Oxford University Press, 1998.

[13] P. Le Chenadec. On the logic of unification, *Journal of Symbolic Computation*, **8**, 141–199, 1989.
[14] T. Leinster. Basic bicategories. *arXiv preprint math,CT/9810017*, 1998.
[15] T, Leinster. *Higher Operads, Higher Categories*, London Mathematical Society Lecture Note Series (Book 298), Cambridge University Press, 2004.
[16] P. L. Lumsdaine. Weak ω-categories from intensional type theory. In *Typed Lambda Calculi and Applications*, Vol. 5608 of *LNCS*, pp. 172–187. Springer, 2009.
[17] D. Prawitz. Inference and knowledge, In *The Logica Yearbook 2008.* pp. 175–192. College Publications, 2009.
[18] T. Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*, https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.
[19] B. van den Berg and R. Garner. Types are weak ω-groupoids, *Proceedings of the London Mathematical Society*. **102**, 370–394, 2011.
[20] V. Voevodsky. *Univalent Foundations and Set Theory*. Univalent Foundations and Set Theory, Lecture at IAS, Princeton, New Jersey, 2014.

Received 31 January 2016

A Subterm substitution

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

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

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

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

where M, N and O are terms.

As proposed in [5], we can define similar rules using computational paths, as follows:

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

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

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

B List of rewrite rules

We present all rewrite rules of $LND_{EQ}-TRS$. They are as follows (All have been taken from [5]):

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