Canonicity of Weak ω-groupoid Laws Using Parametricity Theory

Marc Lasson

Canonicity of weak ω-groupoid laws using parametricity theory

Marc Lasson

INRIA Paris-Rocquencourt, PiR2, Univ Paris Diderot, Sorbonne Paris Cité
F-78153 Le Chesnay France

Abstract

We show that terms witnessing a groupoid law from the ω-groupoid structure of types are all propositionally equal. Our proof reduce this problem to the unicity of the canonical point in the n-th loop space and conclude using Bernardy's parametricity theory for dependent types.

Keywords: type theory, parametricity, loop spaces, groupoids, identity types

1 Introduction

The synthetic approach to weak ω-groupoids promoted by the univalent foundation program [8] is the idea that (homotopy) type theory should be the primitive language in which spaces, points, paths, homotopies are derived. Following this approach, spaces are represented by types, points by inhabitants and paths by equalities between points (also known as identity types) and algebraic properties of these objects should not be enforced *a priori* (for instance by axioms) but should be derived directly from the language. To justify the synthetic approach, one should prove the canonicity of each definition, in the sense that no important choice should be made by choosing a particular implementation of a definition over another.

Garner, van den Berg [9] and Lumsdsaine [5] independently showed that in type theory, each type can be equipped with a structure of weak ω-groupoids. For this, they show that a minimal fragment of Martin-Löf type theory, where identity types are the only allowed type constructors, bears a weak ω-category structure. Informally, these results state the possibility to express algebraic properties of weak ω-groupoids as types and in each case to find a canonical inhabitant of these types reflecting the fact that the property holds. Identities, inversion and concatenation of

[1] Email: marc.lasson@inria.fr

path, associativities, involution of inversion, horizontal and vertical compositions of 2-paths, are all examples of groupoid laws. The *canonicity* of witnesses of groupoid laws here means there is a path between any two inhabitants of the law witnessing their equality, but also that this path should be canonical: there should be a path between any two paths between two inhabitants of the same law, and so on ... This canonicity is already a known fact within the fragment. The main result of this article is to extend the canonicity to the whole Martin-Löf type theory.

In this work, we follow a syntactic approach inspired by Brunerie [3] to formalize the notion of groupoid law. We call *groupoid law* any closed type $\forall \Gamma.c$ such that the sequent $\Gamma \vdash c : \mathsf{Type}$ is derivable in the minimal fragment and such that the context Γ is contractible. A *contractible context* is a context of the following shape: $A : \mathsf{Type}, a : A, x_1 : C_1, y_1 : M_1 = x_1, \ldots, x_n : C_n, y_n : M_n = x_n$ where x_i does not occur in M_i. The shape of these contexts is stable by path-induction, which allows to find an inhabitant of any groupoid law by successive path inductions. We show that this inhabitant is canonical, even outside of the fragment.

The main idea of the proof is to use successive path inductions to reduce the problem of the uniqueness of inhabitants of a given groupoid law to the uniqueness of the canonical point inhabiting a parametric loop space. Given a base type A and a point $a : A$, the n-th *loop space* and its *canonical point* are inductively defined by:

$$\begin{aligned} \Omega_0(A,a) &:= A & \omega_0(A,a) &:= a \\ \Omega_{n+1}(A,a) &:= \Omega_n(a = a, 1_a) & \omega_{n+1}(A,a) &:= \omega_n(a = a, 1_a) \end{aligned}$$

where $1_a : a = a$ denotes the reflexivity. Thus for any integer n, $\forall X : \mathsf{Type}, x : X.\Omega_n(X,x)$ is a groupoid law inhabited by $\lambda X : \mathsf{Type}, x : X.\omega_n(X,x)$ (note that using one universe, it is possible to internalize the quantification over n; everything that we state here will be true whether or not this is used). We call this groupoid law the n-th *parametric loop space*.

The 0-th parametric loop space, is the polymorphic type $\forall X : \mathsf{Type}.X \to X$ of identity functions, and its canonical inhabitant is $\lambda X : \mathsf{Type}, x : X.x$, ie. the identity function. This term is the only one up to function extensionality inhabiting its type. The standard tool to prove this kind of properties is by using Reynold's parametricity theory [7] which was introduced to study the behavior of type quantifications within polymorphic λ-calculus (a.k.a. System F). It refers to the concept that well-typed programs cannot inspect types; they must behave uniformly with respect to abstract types. Reynolds formalizes this notion by showing that polymorphic programs satisfy the so-called logical relations defined by induction on the structure of types. This tool has been extended by Bernardy et al. [2] to dependent type systems. It provides a uniform translation of terms, types and contexts preserving typing (the so-called *abstraction theorem*). In its unary version (the only needed for this work), logical relations are defined by associating to any well-formed type $A : \mathsf{Type}$ a predicate $[\![A]\!] : A \to \mathsf{Type}$ and to any inhabitant $M : A$ a witness $[\![M]\!] : [\![A]\!]\, M$ that the M satisfies the predicate. This translation may be extended to cope with identity types by setting $[\![a = b]\!] : a = b \to \mathsf{Type}$ to be the predicate $\lambda p : a = b.p_*([\![a]\!]) = [\![b]\!]$ where p_* is the transport along p of the predicate gener-

ated by the common type of a and b. Then, it is easy -although quite verbose- to find a translation of introduction and elimination rules of identity types as well as checking that these translations preserve computation rules. This allows to extend Bernardy's abstraction theorem to identity types. Using this framework, we are able to generalize the uniqueness property of the polymorphic identity type to any parametric loop space. The proof proceed by induction on the index of the loop space and uses algebraic properties of transport.

Outline of the paper.

In Section 2, we introduce the type theoretical setting that is used in the article. Section 3 is devoted to the proof that Bernardy's parametricity may be extended to cope with identity types. In Section 4, we use this translation to prove the canonicity result for loop spaces; we prove that all inhabitants of parametric loop spaces are propositionally equal (Theorem 2). In Section 5, we introduce the fragment MLID of type theory to define our notion of groupoid laws and we show that the result of Section 4 may be generalized to all groupoid laws (Theorem 4). Finally Section 6 is devoted to various discussions.

2 Presentation of the syntax

We give a presentation of Martin-Löf type theory with identity types and universes which is close to the syntax of pure type systems [1] in order to reuse the parametricity theory presented in [2]. In this framework, computation rules are treated in an untyped way and subtyping of universes is achieved using Luo's cumulativity relation introduced for the extended calculus of constructions [6]. The reader may be more used to judgemental presentations of type theory where each computation steps are checked to be well-typed. This is just a matter of presentation; all the material presented here could be adapted without much effort to suit type systems using a judgemental equality. Also, the use of cumulativity is not really needed but makes our results more general and closer to implementations such as coq.
The terms of the system are given by the following grammar:

$$\begin{array}{rcl} A,B,C,M,N,U,V := & x \mid (M\,N) \mid \lambda x:A.M \mid \forall x:A.B \mid \mathsf{Type}_i \\ & \mid M =_A N \mid \mathbf{1}^C_M \mid \mathbf{J}_{\forall x:C,y:M=x.A}(B,U,V) \end{array}$$

where universes Type_i are indexed by $i \in \mathbb{N}$. Variables are considered up to α-conversion and we write $M[N/x]$ to denote the term obtained by substituting all free occurrences of x in M by N. To ease the reading of terms, we allow ourselves to omit some typing annotations when they could be guessed from the context (in particular $M = N$, $\mathbf{1}_M$ and $\mathbf{J}(B,U,V)$ will be used often in this text).

The grammar of terms is obtained by adding to the syntax of pure type systems:

- The type constructor $M =_A N$ for forming identity types,
- The introduction rule for identity types, $\mathbf{1}^C_M$, to witness the reflexivity $M =_C M$,

- The elimination rule (also known as "path-induction") $\mathbf{J}_{\forall x:C,y:M=x.A}(B,U,V)$. Given any dependent type $A(x,y)$ which depends on a point x and a path y from a base point M to x, given a witness for $A(M,\mathbf{1}_M)$ (the base case of the induction), given a point U and a path V, the path-induction provides an inhabitant of $A(U,V)$. This formulation of path-induction, due to Paulin-Möhring (also called *based path induction*), is equivalent to the other version where A is parametrized by two points and a path between them.

We use the symbol $\equiv$ to denote the syntactic equality (up to α-conversion) between terms. The conversion between terms will be denoted by $M \equiv_\beta N$, it is defined as the smallest congruence containing the usual β-reduction $(\lambda x : A.M)\, N \equiv_\beta M[N/x]$ and the computation rule for identity types: $\mathbf{J}_{\forall x:c,y:M=x,\Delta.P}(N,M,\mathbf{1}^C_M) \equiv_\beta N$. And the cumulativity order is defined as the smallest partial order $\preccurlyeq$ compatible with $\equiv_\beta$ and satisfying for $i \leq j$:

$$\forall x_1 : A_1, ..., x_n : A_n.\mathsf{Type}_i \preccurlyeq \forall x_1 : A_1, ..., x_n : A_n.\mathsf{Type}_j$$

Like in the extended calculus of constructions, the cumulativity rule is not fully contravariant with respect to the domain of functions ($A' \preccurlyeq A$ and $B \preccurlyeq B'$ does not imply $\forall x : A.B \preccurlyeq \forall x : A'.B'$) otherwise it would break the decidability of type checking. Contexts are finite lists of the form $x_1 : A_1, \cdots, x_n : A_n$ mapping a variable to its type. The rules of the type system are given in Figure 1. As we follow standard lines, we do not develop in details the metatheory of the system.

The non-dependent version of path-induction is called *transport* and is defined by

$$P_*^{\,x:C.X}\,(M) \equiv \mathbf{J}_{\forall x:C,y:U=x.X}(M,V,P)$$

where y does not occur in X. It is often used to coerce between type families: given a type family $X : C \to \mathsf{Type}_i$, a path $P : U = V$ between two points $U, V : C$, and a term M in $X\,U$, the transport $P_*^{\,x:C.X}\,(M)$ of M along P inhabits $X\,V$. It satisfies the following derivable rule :

$$\frac{\Gamma, x : C \vdash X : \mathsf{Type}_j \qquad \Gamma \vdash P : U =_C V \qquad \Gamma \vdash M : X[U/x]}{\Gamma \vdash P_*^{\,x:C.X}\,(M) : X[V/x]}\ \textsc{transport}$$

We sometimes also omit the type family when it can be guessed from the context. The computation rule tells us that transporting along a reflexivity is same as doing nothing : $\mathbf{1}_{U*}(M) \equiv_\beta M$.

3 Extending Relational Parametricity to Identity types

In this section, we explain how to extend Bernardy's parametricity translation below to primitive identity types. Then we prove that this extension preserves typing (The-

$$\frac{}{\langle\rangle\ \mathbf{wf}}\ \text{WF-EMPTY} \qquad \frac{\Gamma\ \mathbf{wf} \qquad \Gamma \vdash B : \mathsf{Type}_i}{\Gamma, x : B\ \mathbf{wf}}\ \text{WF}$$

$$\frac{\Gamma\ \mathbf{wf}}{\Gamma \vdash \mathsf{Type}_i : \mathsf{Type}_{i+1}}\ \text{UNIV} \qquad \frac{\Gamma\ \mathbf{wf} \qquad (x : A) \in \Gamma}{\Gamma \vdash x : A}\ \text{VARIABLES}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : \forall x : A.B}\ \text{ABSTRACTION} \qquad \frac{\Gamma \vdash M : \forall x : A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M\ N) : B[N/x]}\ \text{APPLICATION}$$

$$\frac{\Gamma \vdash M : A' \qquad \Gamma \vdash A : \mathsf{Type}_i}{\Gamma \vdash M : A}\ A' \preccurlyeq A\ \text{CONVERSION} \qquad \frac{\Gamma \vdash A : \mathsf{Type}_i \qquad \Gamma, x : A \vdash B : \mathsf{Type}_i}{\Gamma \vdash \forall x : A.B : \mathsf{Type}_i}\ \text{PRODUCT}$$

$$\frac{\Gamma \vdash M : C \qquad \Gamma \vdash N : C \qquad \Gamma \vdash C : \mathsf{Type}_i}{\Gamma \vdash M =_C N : \mathsf{Type}_i}\ \text{IDENTITY} \qquad \frac{\Gamma \vdash M : C}{\Gamma \vdash \mathbf{1}^C_M : M =_C M}\ \text{REFLEXIVITY}$$

$$\frac{\begin{array}{c}\Gamma \vdash M : C \\ \Gamma, x : C, y : M = x \vdash P : \mathsf{Type}_i\end{array} \qquad \Gamma \vdash B : P[M/x, \mathbf{1}^C_M/y] \qquad \begin{array}{c}\Gamma \vdash U : C \\ \Gamma \vdash V : M = U\end{array}}{\Gamma \vdash \mathbf{J}_{\forall x:C, y:M=x.P}(B, U, V) : P[U/x, V/y]}\ \text{PATH-INDUCTION}$$

Fig. 1. Type theory with identity types (MLTT).

orem 1).

$$\begin{aligned}
[\![\forall x : A.B]\!] &\equiv \lambda f : \forall x : A.B. \forall x : A, x_R : x \in [\![A]\!].(f\ x) \in [\![B]\!] \\
[\![\mathsf{Type}_i]\!] &\equiv \lambda x : \mathsf{Type}_i. x \to \mathsf{Type}_i \\
[\![\lambda x : A.M]\!] &\equiv \lambda x : A, x_R : x \in [\![A]\!]. [\![M]\!] \\
[\![M\ N]\!] &\equiv [\![M]\!]\ N\ [\![N]\!] \\
[\![x]\!] &\equiv x_R
\end{aligned}$$

where $M \in [\![A]\!]$ simply stands as a notation for $[\![A]\!]\ M$ (it makes formulas a bit easier to read). As said in the introduction, the predicate generated by an identity type $M =_C N$ is defined by the type family over $M =_C N$ selecting paths transporting $[\![M]\!]$ to $[\![N]\!]$:

$$[\![M =_C N]\!] : M =_C N \to \mathsf{Type}$$
$$[\![M =_C N]\!] \equiv \lambda p : M =_C N. {p_*}^{x:C.x \in [\![C]\!]}\ ([\![M]\!]) =_{[\![C]\!]\ N} [\![N]\!]$$

Thanks to the computational rule

$$\mathbf{1}^C_M \in [\![M =_C M]\!] \quad \equiv_\beta \quad {\mathbf{1}^C_{M*}}^{x:c.x \in [\![C]\!]}\ ([\![M]\!]) =_{M \in [\![C]\!]} [\![M]\!] \quad \equiv_\beta \quad [\![M]\!] =_{M \in [\![C]\!]} [\![M]\!]$$

the translation $[\![\mathbf{1}^C_M]\!] : \mathbf{1}^C_M \in [\![M =_C M]\!]$ of reflexivity is a reflexivity: $[\![\mathbf{1}^C_M]\!] \equiv \mathbf{1}^{M \in [\![C]\!]}_{[\![M]\!]}$. And finally, the elimination rule is translated in terms of nested path-

inductions:

$$\begin{aligned}&[\![\mathbf{J}_{\forall x:C,y:M=x.P}(B,U,V)]\!] \equiv \\ &\quad \mathbf{J}_{\forall x_R:U\in[\![C]\!],y_R:V_*([\![M]\!])=x_R.\mathbf{J}_{\forall x:C,y:M=x.P}(B,U,V)\in[\![P]\!][U/x,V/y]} \\ &\qquad (\mathbf{J}_{\forall x:C,y:M=x.\mathbf{J}_{\forall x:C,y:M=x.P}(B,x,y)\in[\![P]\!][y_*([\![M]\!])/x_R,\mathbf{1}_{y_*([\![M]\!])}/y_R]}([\![B]\!],U,V),[\![U]\!],[\![V]\!])\end{aligned}$$

The translation of the predicate of path inductions are quite verbose and make the translation hard to read. However if we ignore the annotation, we see that the translation $[\![\mathbf{J}(B,U,V)]\!] \equiv \mathbf{J}(\mathbf{J}([\![B]\!],U,V),[\![U]\!],[\![V]\!])$ is simply a duplication of path induction which should be compared to the duplication in abstractions and applications. We can check that this translation behaves well with respect to substitution and conversion :

Lemma 1 (Substitution and conversion lemma)**.** *We have :*

(i) $[\![M[N/x]]\!] \equiv [\![M]\!][N/x,[\![N]\!]/x_R]$,

(ii) *If* $M \equiv_\beta M'$, *then* $[\![M]\!] \equiv_\beta [\![M']\!]$.

(iii) *If* $M \preccurlyeq M'$, *then* $[\![M]\!] \preccurlyeq [\![M']\!]$.

Proof. The proof of (i) is a routine proof by induction on M. And (iii) is a rather direct consequence of (ii). To prove (ii), we only do here the only check that is not in Bernardy's translation :

$$\begin{aligned}[\![\mathbf{J}(N,M,\mathbf{1}^C_M)]\!] &\equiv \mathbf{J}(\mathbf{J}([\![N]\!],M,\mathbf{1}^C_M),[\![M]\!],[\![\mathbf{1}^C_M]\!]) \\ &\equiv \mathbf{J}(\mathbf{J}([\![N]\!],M,\mathbf{1}^C_M),[\![M]\!],\mathbf{1}^{M\in[\![C]\!]}_{[\![M]\!]}) \\ &\equiv_\beta \mathbf{J}([\![N]\!],[\![M]\!],\mathbf{1}^{M\in[\![C]\!]}_{[\![M]\!]}]\!]) \\ &\equiv_\beta [\![N]\!] \qquad\qquad \square\end{aligned}$$

Now we can check that this extension preserve typing :

Theorem 1 (Abstraction)**.** *If* $\Gamma \vdash M : A$, *then*

$$\begin{cases}[\![\Gamma]\!] \vdash M : A & (a) \\ [\![\Gamma]\!] \vdash [\![M]\!] : M \in [\![A]\!] & (b)\end{cases}$$

moreover if Γ **wf** *then* $[\![\Gamma]\!]$ **wf** *(c).*

Proof. The abstraction theorem is proved by induction on derivations. In each cases, proving the statement (a) is straightforward. Since the treatment of other rules is now standard, we only deal here with the rules concerning identity types. Even though we do not detailed it here, the treatment of CONVERSION uses previous lemma.

- IDENTITY: The induction hypothesis gives us $[\![\Gamma]\!] \vdash [\![M]\!] : M \in [\![C]\!]$ (1), $[\![\Gamma]\!] \vdash [\![N]\!] : N \in [\![C]\!]$ (2) and $[\![\Gamma]\!], x : C \vdash x \in [\![C]\!] : \mathsf{Type}_i$ (3). Using TRANSPORT we derive from (1) and (3), that $[\![\Gamma]\!], p : M =_C N \vdash p_*^{\lambda x:C.x\in C}([\![M]\!]) : N \in [\![C]\!]$. Then, using IDENTITY and (2) we build a derivation for $[\![\Gamma]\!], p : M =_C N \vdash$

$p_*{}^{\lambda x:C.x\in C}([\![M]\!]) =_{N\in[\![C]\!]} [\![N]\!]$ and finally we conclude $[\![\Gamma]\!] \vdash [\![M =_C N]\!] : M =_C N \to \mathsf{Type}_i$ with ABSTRACTION.

- REFLEXIVITY: By induction hypothesis, we have $[\![\Gamma]\!] \vdash [\![M]\!] : M \in [\![C]\!]$ (1). Using (1) we can check that $[\![\Gamma]\!] \vdash \mathbf{1}^{M\in[\![C]\!]}_{[\![M]\!]} : [\![M]\!] =_{M\in[\![C]\!]} [\![M]\!]$ which is convertible to $[\![\Gamma]\!] \vdash \mathbf{1}^{M\in[\![C]\!]}_{[\![M]\!]} : \mathbf{1}^C_{M*}([\![M]\!]) =_{M\in[\![C]\!]} [\![M]\!]$. So, we conclude $[\![\Gamma]\!] \vdash [\![\mathbf{1}^C_M]\!] : \mathbf{1}^C_M \in [\![C]\!]$.
- PATH-INDUCTION: The induction hypothesis for (b) are :

$$[\![\Gamma]\!] \vdash [\![M]\!] : M \in [\![C]\!] \quad (1) \qquad [\![\Gamma, x : C, y : M = x]\!] \vdash [\![P]\!] : P \to \mathsf{Type}_i \quad (2)$$
$$[\![\Gamma]\!] \vdash [\![B]\!] : B \in [\![P[M/x, \mathbf{1}^C_M/y]]\!] \quad (3) \qquad [\![\Gamma]\!] \vdash [\![U]\!] : U \in [\![C]\!] \quad (4)$$
$$[\![\Gamma]\!] \vdash [\![V]\!] : V \in [\![M = U]\!] \quad (5)$$

and the induction hypothesis for (a) are :

$$[\![\Gamma]\!] \vdash M : C \quad (6) \qquad [\![\Gamma, x : C, y : M = x]\!] \vdash P : \mathsf{Type}_i \quad (7)$$
$$[\![\Gamma]\!] \vdash B : P[M/x, \mathbf{1}^C_M/y] \quad (8) \qquad [\![\Gamma]\!] \vdash U : C \quad (9)$$
$$[\![\Gamma]\!] \vdash V : M = U \quad (10)$$

Let T be the following term :

$$\mathbf{J}_{\forall x:C, y:M=x.\mathbf{J}(B,x,y)\in[\![P]\!][y_*([\![M]\!])/x_R, \mathbf{1}_{y_*([\![M]\!])}/y_R]}([\![B]\!], U, V)$$

First, we have to typecheck T by showing that

$$[\![\Gamma]\!] \vdash T : \mathbf{J}(B, U, V) \in [\![P]\!][y_*([\![M]\!])/x_R, \mathbf{1}_{y_*([\![M]\!])}/y_R, U/x, V/y] \qquad (*)$$

which means, using PATH-INDUCTION, checking :

· The predicate is well-formed :

$$[\![\Gamma]\!], x : C, y : M = x \vdash \mathbf{J}(B, x, y) \in [\![P]\!][y_*([\![M]\!])/x_R, \mathbf{1}_{y_*([\![M]\!])}/y_R] : \mathsf{Type}$$

This is obtained by substituting x_R and y_R in (2) using a derivation of

$$[\![\Gamma]\!], x : C, y : M = x \vdash y_*([\![M]\!]) : x \in [\![C]\!]$$

and of $[\![\Gamma]\!], x : C, y : M = x \vdash \mathbf{1}_{y_*([\![M]\!])} : y_*([\![M]\!]) = y_*([\![M]\!])$ which are easily derived from (1) and by applying $[\![\Gamma]\!], x : C, y : M = x \vdash \mathbf{J}(B, x, y) : P$ to the substituted derivation.

· The arguments are correct : We derive

$$[\![\Gamma]\!] \vdash [\![B]\!] : \mathbf{J}(B, M, \mathbf{1}_M) \in [\![P]\!][y_*([\![M]\!])/x_R, \mathbf{1}_{y_*([\![M]\!])}/y_R, M/x, \mathbf{1}_M/y]$$

by noticing using computation rules that :

$$\begin{aligned}\mathbf{J}(B, M, \mathbf{1}_M) &\in [\![P]\!][y_*([\![M]\!])/x_R, \mathbf{1}_{y_*([\![M]\!])}/y_R, M/x, \mathbf{1}_M/y] \\ &\equiv_\beta B \in [\![P]\!][M/x, [\![M]\!]/x_R, \mathbf{1}_M/y, \mathbf{1}_{y_*([\![M]\!])}/y_R]\end{aligned}$$

and therefore CONVERSION and (3) allow us to conclude. Finally, we have to check that target point and path are correct ie. $[\![\Gamma]\!] \vdash U : C$ and $[\![\Gamma]\!] \vdash V : M = U$ which are given by (9) and (10).

Let K be the following term:

$$\mathbf{J}_{\forall x_R : U \in [\![C]\!], y_R : V_*([\![M]\!]) = x_R . \mathbf{J}_{\forall x:C, y:M=x.P}(B,U,V) \in [\![P]\!][U/x, V/y]}(T, [\![U]\!], [\![V]\!])$$

We now want to check that : $[\![\Gamma]\!] \vdash K : \mathbf{J}_{\forall x:C, y:M=x.P}(B, U, V) \in [\![P[U/x, V/y]]\!]$ using PATH-INDUCTION, we have to show that :

· The predicate is well-formed :

$$[\![\Gamma]\!], x_R : U \in [\![C]\!], y_R : V_*([\![M]\!]) = x_R \vdash \\ \mathbf{J}_{\forall x:C, y:M=x.P}(B, U, V) \in [\![P]\!][U/x, V/y] : \mathsf{Type}_i$$

which is obtained by substituting x and y in (2) using (9) and (10) and by applying $[\![\Gamma]\!] \vdash \mathbf{J}_{\forall x:C, y:M=x.P}(B, U, V) : P[U/x, V/y]$ to the substituted derivation.

· The arguments are correct : We use (*) for type checking T and we use (4) and (5) for type checking $[\![U]\!]$ and $[\![V]\!]$. □

4 Canonicity in parametric loop spaces

The goal of this section is to prove Theorem 2. The following lemma is needed to perform the main induction in Lemma 3. The proofs terms are described in a semi-formal style for reading purpose, they could be easily constructed from the prose. However, as a consequence of Thereom 2, the precise shape of these terms is not important. In this section, we use $p \cdot q$ and p^{-1} to denote respectively the concatenation of paths and the inverse.

Lemma 2. *Let $P : A \to \mathsf{Type}$ be a type family and $u : P\,a$ for some $a : A$ and assume we have $\phi : \forall x : A. P\,x \to a = x$. Then, for all $p : a = a$ such that $p_*(u) = u$, we have $1 = p$.*

Proof. By applying the functorial action of path on $p_*(u) = u$ using ϕ_a we obtain a two dimensional path $\phi_a(p_*(u)) = \phi_a(u)$ and by the naturality of transport (Lemma 2.3.11 in [8]), we obtain a path $p_*(\phi_a(u)) = \phi_a(u)$. The left-hand side path $p_*(\phi_a(u))$ is transport over an identity type, we therefore have $p_*(\phi_a(u)) = \phi_a(u) \cdot p$ and therefore $\phi_a(u) \cdot p = \phi_a(u)$. So, we conclude $1 = p$ by cancelling by $\phi_a(u)$ and by symmetry. □

Here is the main induction which allows us to derive Theorem 2:

Lemma 3. *Let A, P, a, u and ϕ defined as in previous lemma. Then for all n-dimensional loop $p : \Omega_n(A, a)$, the type family $[\![\Omega_n]\!](A, P, a, u) : \Omega_n(A, a) \to \mathsf{Type}$ satisfies : $\forall p : \Omega_n(A, a).[\![\Omega_n]\!](A, P, a, u)\,p \to \omega_n(A, a) = p$.*

Proof. By induction on n, the base case is exactly witness by ϕ. For the inductive step, we need to prove $\forall p : \Omega_{n+1}(A, a).p \in [\![\Omega_{n+1}]\!](A, P, a, u) \to \omega_{n+1}(A, a) = p$

which is convertible to :

$$\forall p : \Omega_n(a = a, 1).p \in [\![\Omega_n]\!](a = a, \lambda q : a = a.q_*(u) = u, 1, 1) \to \omega_n(a = a, 1) = p$$

We may therefore apply the induction hypothesis by providing a proof that $\forall q : a = a.q_*(u) = u \to 1 = q$ which is given by previous lemma. □

We can deduce :

Theorem 2 (Canonicity for loop spaces)**.** *If* $\vdash M : \forall A : \mathsf{Type}, a : A.\Omega_n(A, a)$ *then there is a term* π *such that* $\vdash \pi : \forall A : \mathsf{Type}, a : A.\omega_n(A, a) = M\, A\, a$.

Proof. The abstraction theorem gives us a proof $[\![M]\!]$ that $\forall A : \mathsf{Type}, A_R : A \to \mathsf{Type}, a : A, a_R : A_R\, a.(M\, A\, a) \in [\![\Omega_n]\!](A, A_R, a, a_r)$ by instantiating A_R with $\lambda x : A.a = x$ we can conclude by applying previous lemma. □

Note that as a corollary, the proof π is unique up to propositional equality (and so on). If we have two such proofs π and π' then $(\pi\, A\, a)\cdot(\pi'\, A\, a)^{-1}$ is of type $\omega_n(A, a) = \omega_n(A, a)$ which is $\Omega_{n+1}(A, a)$. Therefore, by applying the previous theorem we obtain a proof of $(\pi\, A\, a)\cdot(\pi'\, A\, a)^{-1} = \omega_{n+1}(A, a)$ which is also $(\pi\, A\, a)\cdot(\pi'\, A\, a)^{-1} = \mathbf{1}_{\omega_n(A,a)}$. Therefore we conclude $(\pi\, A\, a) = (\pi'\, A\, a)$.

5 Groupoid laws

In this section, we start by describing a fragment MLID of the previous type system MLTT. This sub-system is used to characterise groupoid laws. Informally, it is obtained from MLTT by removing the rules ABSTRACTION, APPLICATION and UNIVERSES and by restricting valid sequents to the contractible contexts defined in the introduction. In the absence of function spaces, the rule PATH-INDUCTION has to be strengthened in order to be able to make a path induction along a path which is not the last one of the context. Therefore, we need to extend the grammar of terms in MLID with terms of the shape $\mathbf{J}_{\forall x:C, y:M=x, \Delta.P}(B, U, V, \overrightarrow{W})$ where Δ is a context and where the vectorial notation $\overrightarrow{W}$ denotes a tuple $(W_1, ..., W_n)$ of terms.

The typing rules are given in Figure 2. Formally a context Γ is said to be *contractible* if Γ **contr** is derivable; the reader should notice that all contexts occurring in derivations of MLID are contractible. In the typing rules, we write $\Gamma \vdash \overrightarrow{W} : \Delta$ to denote the conjunction for $k = 1, \cdots, n$ of $\Gamma \vdash W_k[A_1/y_1, \cdots, A_{k-1}/y_{k-1}] : A_k$ when Δ is of the shape $y_1 : A_1, \cdots, y_n : A_n$. Moreover, $[\overrightarrow{W}/\Delta]$ denotes the iterated substitution $[W_1/y_1, \cdots, W_n[A_1/y_1, \cdots, A_{n-1}/y_{n-1}]/y_n]$.

In order to embed MLID into MLTT, we translate extended path inductions into normal ones according to the following translation :

$$\begin{aligned}&\mathbf{J}_{\forall x:C, y:M=x, \Delta.P}(B, U, V, \overrightarrow{W}) \equiv \\ &\quad \lambda \overrightarrow{x} : \Delta.(\mathbf{J}_{\forall x:C, y:M=x.\forall \Delta.P}(\lambda x : \Delta[M/x, \mathbf{1}^C_M/y].B, U, V)\ \overrightarrow{x})\end{aligned}$$

where $\overrightarrow{y} : \Delta$ means that $y_1, \cdots, y_n$ are the variables assigned in Δ, and where $\forall\Delta$, $\lambda\Delta$, and $(J\ \overrightarrow{x})$ denote respectively iterated products, abstractions and appli-

$$\frac{}{A : \mathsf{Type}_0, x : A \ \mathbf{contr}}\ \textsc{init} \qquad \frac{\Gamma \ \mathbf{contr} \qquad \Gamma \vdash_{\mathrm{id}} B : \mathsf{Type}_0 \qquad \Gamma \vdash_{\mathrm{id}} M : B}{\Gamma, x : B, p : M =_B x \ \mathbf{contr}}\ \textsc{contr}$$

$$\frac{\Gamma \vdash_{\mathrm{id}} M : C \qquad \Gamma \vdash_{\mathrm{id}} N : C \qquad \Gamma \vdash_{\mathrm{id}} C : \mathsf{Type}_0}{\Gamma \vdash_{\mathrm{id}} M =_C N : \mathsf{Type}_0}\ \textsc{identity} \qquad \frac{\Gamma \vdash_{\mathrm{id}} M : C}{\Gamma \vdash_{\mathrm{id}} \mathbf{1}^C_M : M =_C M}\ \textsc{reflexivity}$$

$$\frac{\Gamma \ \mathbf{contr} \qquad (x : A) \in \Gamma}{\Gamma \vdash_{\mathrm{id}} x : A}\ \textsc{variable} \qquad \frac{\Gamma \vdash_{\mathrm{id}} M : A' \qquad \Gamma \vdash_{\mathrm{id}} A' : \mathsf{Type}_0}{\Gamma \vdash_{\mathrm{id}} M : A}\ A' \equiv_\beta A\ \textsc{conversion}$$

$$\frac{\begin{array}{c}\Gamma \vdash_{\mathrm{id}} M : C \qquad\qquad \Gamma \vdash_{\mathrm{id}} U : C \\ \Gamma, x : C, y : M = x, \Delta \vdash_{\mathrm{id}} P : \mathsf{Type}_0 \qquad \Gamma \vdash_{\mathrm{id}} V : M = U \\ \Gamma, \Delta[M/x, \mathbf{1}^C_M/y] \vdash_{\mathrm{id}} B : P[M/x, \mathbf{1}^C_M/y] \qquad \Gamma \vdash_{\mathrm{id}} \overrightarrow{W} : \Delta\end{array}}{\Gamma \vdash_{\mathrm{id}} \mathbf{J}_{\forall x:C, y:M=x, \Delta.P}(B, U, V, \overrightarrow{W}) : P[U/x, V/y, \overrightarrow{W}/\Delta]}\ \textsc{extended-path-induction}$$

Fig. 2. The minimal fragment MLID of type theory with identity types.

cations. It is then straightforward to check that EXTENDED-PATH-INDUCTION is an admissible rule of MLTT.

Groupoid laws are characterized by a contractible context and a derivable type in MLID (Figure 3 contains some examples of groupoid laws):

Definition (Groupoid law)**.** *A* groupoid law *is a term of the shape* $\forall\Gamma.C$ *such that* $\Gamma \vdash_{\mathrm{id}} C : \mathsf{Type}_0$.

The only groupoid laws in the "initial" contractible context $A : \mathsf{Type}, a : A$ are loop spaces. Since we do not change the base type and the base point of loop spaces in this section we simply denote by ω_n (resp. Ω_n) the terms $\omega_n(A, a)$ (resp. $\Omega_n(A, a)$). Using these notations we have :

Lemma 4. *If* $A : \mathsf{Type}, a : A \vdash_{\mathrm{id}} T : \mathsf{Type}_0$*, then*

(i) there exists $|T| \in \mathbb{N}$ *such that* $T \equiv_\beta \Omega_{|T|}$*,*

(ii) moreover, if $A : \mathsf{Type}, a : A \vdash_{\mathrm{id}} W : T$ *then* $W \equiv_\beta \omega_{|T|}$*.*

Proof. We proceed by induction on the derivation of $A : \mathsf{Type}, a : A \vdash_{\mathrm{id}} T : \mathsf{Type}_0$. We notice that the only two possible last used rules are IDENTITY and VARIABLE since Type_0 does not inhabit Type_0 it is not possible to invoke EXTENDED-PATH-INDUCTION nor CONVERSION.

(i) In the VARIABLE case, we necessarily have $T \equiv A$ and therefore we can conclude by taking $|T| = 0$. In the IDENTITY case, T is of the shape $M =_C N$ and by induction hypothesis $M \equiv_\beta \omega_{|C|}$, $N \equiv_\beta \omega_{|C|}$ and $C \equiv_\beta \Omega_{|C|}$. So we conclude by taking $|T| = |C| + 1$.

(ii) Without loss of generality (MLTT is normalizing) we can assume that W is in normal form. We proceed by a (nested) induction on the derivation $A :$

$$\begin{aligned}
\texttt{refl} &: \forall X : \mathsf{Type}, x : X. x = x \qquad \texttt{sym} : \forall X : \mathsf{Type}, x : X. x = y \to y = x \\
\texttt{concat} &: \forall X : \mathsf{Type}, x : X, y : X. x = y \to \forall z : X. y = z \to x = z \\
\texttt{assoc} &: \forall X : \mathsf{Type}, x : X, y : X, p : x = y, z : X, q : y = z, t : X, r : z = t. \\
&\qquad \texttt{concat}\, X\, x\, z\, (\texttt{concat}\, X\, x\, y\, p\, z\, q)\, t\, r = \texttt{concat}\, X\, x\, y\, p\, t\, (\texttt{concat}\, X\, y\, z\, q\, t\, r) \\
\texttt{neutral} &: \forall X : \mathsf{Type}, x : X, y : X, p : x = y. (\texttt{concat}\, X\, x\, y\, p\, y\, (\texttt{refl}\, X\, y)) = p \\
\texttt{idem} &: \forall X : \mathsf{Type}, x : X, y : X, p : x = y. \texttt{sym}\, X\, y\, x\, (\texttt{sym}\, X\, x\, y\, p) = p \\
\texttt{horizontal} &: \forall X : \mathsf{Type}, x : X, y : X, p : x = y, p' : x = y. p = p' \to \\
&\qquad \forall z : X, q : x = z, q' : x = z. q = q' \to \texttt{concat}\, X\, x\, y\, p\, z\, q = \texttt{concat}\, X\, x\, y\, p'\, z\, q'
\end{aligned}$$

Fig. 3. Examples of groupoid laws with aliases for canonical inhabitants

$\mathsf{Type}, a : A \vdash_{\mathrm{id}} W : T$, we treat each possible case (IDENTITY and CONVERSION are obviously impossible since T cannot be Type_0):

- VARIABLE: In this case, we necessarily have $W \equiv a$ and $T \equiv A$. So we have $|T| = 0$ and $W \equiv \omega_0$.
- REFLEXIVITY: In this case, W and T are respectively of the shape $\mathbf{1}^C_M$ and $M =_C M$. By induction hypothesis, we have $M \equiv_\beta \omega_{|C|}$. Therefore $|T| = |C| + 1$ and $W \equiv_\beta \omega_{|C|+1}$.
- EXTENDED-PATH-INDUCTION: This is in fact an impossible case. We would have W of the shape $\mathbf{J}_{\forall x:C, y:M=x, \Delta.P}(B, U, V, \overrightarrow{Z})$ and by induction hypothesis, we would have $M \equiv_\beta \omega_{|C|} \equiv_\beta U$ and $V \equiv_\beta \omega_{|C|+1}$. Therefore V is a reflexivity and so W is not in normal form. □

The previous lemma allow us to find a canonic instantiation of any contractible context given by :

$$\begin{aligned}
(A : \mathsf{Type}, a : A)^+ &= (A, a) \\
(\Gamma, x : A, y : M =_C x)^+ &= (\Gamma^+, \omega_{|C[\Gamma^+/\Gamma]|}, \omega_{|C[\Gamma^+/\Gamma]|+1})
\end{aligned}$$

The following lemma state that this instantiation is correct :

Lemma 5.

$$\Gamma \ \mathbf{contr} \ \ \textit{implies} \ \ A : \mathsf{Type}_0, a : A \vdash_{\mathrm{id}} \Gamma^+ : \Gamma \tag{1}$$

$$\Gamma \vdash_{\mathrm{id}} T : \mathsf{Type}_0 \ \ \textit{implies} \ \ T[\Gamma^+/\Gamma] \equiv_\beta \Omega_{|T[\Gamma^+/\Gamma]|} \tag{2}$$

$$\Gamma \vdash_{\mathrm{id}} T : \mathsf{Type}_0 \ \textit{and} \ \Gamma \vdash_{\mathrm{id}} M : T \ \ \textit{implies} \ \ M \equiv_\beta \omega_{|T[\Gamma^+/\Gamma]|} \tag{3}$$

Proof. We proceed by induction on size of derivations in MLID. We prove (1) by inspecting the last possible rule :

- INITIAL: Γ is of the shape $A : \mathsf{Type}_0, a : A$ and $\Gamma^+ = (A, a)$. By using two times VARIABLE, we check that : $(A : \mathsf{Type}_0, a : A) \vdash_{\mathrm{id}} (A, a) : (A : \mathsf{Type}_0, a : A)$.
- CONTRACTIBLE: Γ is of the shape $\Delta, x : B, p : M =_B x$ with $\Delta \vdash_{\mathrm{id}} B : \mathsf{Type}_0$ and $\Gamma \vdash_{\mathrm{id}} M : B$. By induction hypothesis, we have $A : \mathsf{Type}_0, a : A \vdash_{\mathrm{id}} \Delta^+ : \Delta$, $B[\Delta^+/\Delta] \equiv_\beta \Omega_{|B[\Delta^+/\Delta]|}$ and $M[\Delta^+/\Delta] \equiv_\beta \omega_{|B[\Delta^+/\Delta]|}$. It is then easy to check using CONVERSION that we have $(A : \mathsf{Type}_0, a : A) \vdash_{\mathrm{id}} (\Delta^+, \omega_{|C[\Gamma^+/\Gamma]|}, \omega_{|c[\Gamma^+/\Gamma]|+1}) :$

$(\Delta, x : B, p : M =_B x)$.

To prove (2) and (3), we notice that within the derivation of $\Gamma \vdash_{\mathrm{id}} T : \mathsf{Type}_0$ there is a strictly smaller derivation of Γ **contr** so we can use the induction hypothesis to obtain $A : \mathsf{Type}_0, a : A \vdash_{\mathrm{id}} \Gamma^+ : \Gamma$. Now by substitution, we derive that $A : \mathsf{Type}_0, a : A \vdash_{\mathrm{id}} T[\Gamma^+/\Gamma] : \mathsf{Type}_0$ and $A : \mathsf{Type}_0, a : A \vdash_{\mathrm{id}} M[\Gamma^+/\Gamma] : T[\Gamma^+/\Gamma]$. We conclude that $T[\Gamma^+/\Gamma] \equiv_\beta \Omega_{|T[\Gamma^+/\Gamma]|}$ and $M \equiv_\beta \omega_{|T[\Gamma^+/\Gamma]|}$ by previous lemma. □

Lemma 6 (All groupoid laws are inhabited)**.** *If* $\Gamma \vdash_{\mathrm{id}} T : \mathsf{Type}_0$*, then there exists* $\theta_{\Gamma.T}$ *such that* $\Gamma \vdash_{\mathrm{id}} \theta_{\Gamma.T} : T$.

Proof. The contractible context Γ is of the shape

$$A : \mathsf{Type}_0, a : A, x_1 : C_1, y_1 : M_1 = x_1, \ldots, x_n : C_n, y_n : M_n = x_n$$

we construct $\theta_{\Gamma.c}$ by n successive extended path-inductions (from left to right). After all the inductions, it remains to find an inhabitant of $T[\Gamma^+/\Gamma]$ in the context $A : \mathsf{Type}_0, a : A$. But thanks to the previous lemma, we know that $T[\Gamma^+/\Gamma] \equiv_\beta \Omega_{|T[\Gamma^+/\Gamma]|}$, therefore using CONVERSION we can use $\omega_{|T[\Gamma^+/\Gamma]|}$. Spelled out, the term $\theta_{\Gamma.T}$ is :

$$\begin{aligned}\theta_{\Gamma.T} \equiv \mathbf{J}_{\forall x_1:C_1, y_1:M_1=x_1, \ldots, x_n:C_n, y_n:M_n=x_n.T}(&\\ \mathbf{J}_{\forall (x_2:C_2, y_2:M_2=x_2, \ldots, x_n:C_n, y_n:M_n=x_n.T)[\omega_{|C_1|}/x_1, \omega_{|C_1|+1}/y_1]}(&\\ \cdots \mathbf{J}_{\forall (x_n:C_n, y_n:M_n=x_n.T)[\Delta^+/\Delta]}(\omega_{|T[\Gamma^+/\Gamma]|}, x_n, y_n) \cdots, x_2, y_2), x_1, y_1)\end{aligned}$$

where Δ is the context such that $\Gamma = \Delta, x_n : C_n, y_n : M_n = x_n$. □

As a corollary, we obtain the following theorem :

Theorem 3 (Canonicity for groupoid laws in MLID)**.** *If* $\Gamma \vdash_{\mathrm{id}} T : \mathsf{Type}$ *and if we have two terms* M *and* N *such that* $\Gamma \vdash_{\mathrm{id}} M : T$ *and* $\Gamma \vdash_{\mathrm{id}} N : T$*, then there is a proof* π *such that* $\Gamma \vdash \pi : M = N$.

Proof. Simply notice that $\Gamma \vdash M = N : \mathsf{Type}_0$ and apply previous lemma. □

The reader should remark here that the proof π is also unique up to equality (by applying the theorem to $M = N$!). We are now ready to show that previous theorem may be generalized to the whole system MLTT by using parametricity theory.

Theorem 4 (Canonicity of inhabitants of groupoid laws in MLTT)**.** *If* $\Gamma \vdash_{\mathrm{id}} T : \mathsf{Type}$*,* $\vdash M : \forall \Gamma.T$ *and* $\vdash N : \forall \Gamma.T$*, then there is a proof* π *such that* $\overrightarrow{\gamma} : \Gamma \vdash \pi : (M\ \overrightarrow{\gamma}) = (N\ \overrightarrow{\gamma})$.

Proof. By successive extended path-inductions (from left to right), we can derive $\overrightarrow{\gamma} : \Gamma \vdash (M\ \overrightarrow{\gamma}) = (N\ \overrightarrow{\gamma})$ from a derivation of $A : \mathsf{Type}_0, a : A \vdash (M\,\Gamma^+) = (N\,\Gamma^+)$. We notice that the type of $(M\,\Gamma^+)$ is $T[\Gamma^+/\Gamma]$ which is typable in MLID; by substitution we have $A : \mathsf{Type}_0, a : A \vdash_{\mathrm{id}} T[\Gamma^+/\Gamma] : \mathsf{Type}_0$. Therefore Lemma 5 gives us that $T[\Gamma^+/\Gamma] \equiv_\beta \Omega_n$ for some $n \in \mathbb{N}$. Using CONVERSION, we have $A : \mathsf{Type}_0, a : A \vdash (M\,\Gamma^+) : \Omega_n$. Therefore $(M\,\Gamma^+)$ is an inhabitant of a parametric loop space; therefore we can invoke Theorem 2 to obtain of proof that $(M\,\Gamma^+) = \omega_n$. Similarly

we have a proof of $(N\,\Gamma^+) = \omega_n$ and by concatenating them we obtain a proof of $(M\,\overrightarrow{\gamma}) = (N\,\overrightarrow{\gamma})$. □

Finally, using the same arguments as at the end of Section 4 we can prove that π is unique up to propositional equality (and so on).

6 Discussions

6.1 The definition of groupoid laws

Our definition of MLID is inspired by an unpublished note written by Brunerie [3]. Our syntax for contractible contexts is a bit more general: in Brunerie's definition the starting point of paths occurring at odd positions are always variable (ie. M_k is always a variable) and there is no computation rules in his syntax. Brunerie defines ω-groupoids as models of its syntax, since the absence of computation rules makes his framework more free about how coherence issues are dealt with. However, the goal of our syntax is not give the general syntax for weak ω-groupoids but rather to study only the groupoid structure in the particular case of type theory where computation rules are the natural way to deal with coherence. Nevertheless, it would be an interesting future work to make a precise comparison between other definitions of ω-groupoids and models of MLID.

The semantical nature of Garner and van den Berg [9] makes it quite difficult to relate to our work, however we believe that Lumsdaine's construction [5] of a contractible globular operad may be described in our framework.

6.2 The n-ary case

Throughout this article, we only use the unary case of parametricity theory, but it could be easily generalized to the binary case by transporting along two paths : $[\![x = y]\!]_2 \equiv \lambda(p : x = y)(q : x' = y').p_*(q_*(x_R)) = y_R$. This translation is well-typed under the binary translation $[\![\alpha : \mathsf{Type}_i, x : \alpha, y : \alpha]\!]_2$ given by

$$\alpha\,\alpha' : \mathsf{Type}_i, \alpha_R : \alpha \to \alpha' \to \mathsf{Type}_i, x : \alpha, x' : \alpha', x_R : \alpha_R\,x\,x', y : \alpha, y' : \alpha', y_R : \alpha_R\,y\,y'$$

The translation $[\![\mathbf{1}^\alpha_x]\!]_2 : \mathbf{1}^\alpha_{x*}\big(\mathbf{1}^\alpha_{x'*}(x_R)\big) = x_R$ of $\mathbf{1}^\alpha_x$ is given by $[\![\mathbf{1}^\alpha_x]\!]_2 \equiv \mathbf{1}^{\alpha_R\,x\,x}_{x_R}$ which is defined under the translated context $[\![\alpha : \mathsf{Type}, x : \alpha]\!]_2$. Similarly, the translation of path-induction is obtained by nesting 2+1 path-induction:

$$[\![\mathbf{J}(B, U, V)]\!]_2 \equiv \mathbf{J}(\mathbf{J}(\mathbf{J}([\![B]\!]_2, U, V), U', V'), [\![U]\!]_2, [\![V]\!]_2)$$

It is a routine check to generalize the abstraction theorem of Section 3, in order to a have a binary version of parametricity. Likewise, the n-ary case, is obtained by transporting along n path and the translation of path-induction is obtained by nesting $n + 1$ path-inductions.

6.3 Encoding identity types with inductive families.

In dependent type systems that support inductive families, it is possible to encode identity types by an inductive predicate [4]. For instance, in the coq proof assistant:

```
Inductive paths (A : Type) (a : A) : A → Type := idpath : paths A a a.
```

As explained in [2] (Section 5.4), the parametricity translation extends well to inductive families. The idea is to translate an inductive type I by a new inductive I_R whose constructors are the translation of constructors of I. Likewise, the elimination scheme for I_R is the translation of the one of I.

```
Inductive paths_R (A : Type) (A_R : A → Type) (a : A) (a_R : A_R a) :
    forall x, A_R x → paths A a x → Type :=
  idpath_R : paths_R A A_R a a_R a a_R (idpath A a).
```

There is an equivalence of types (in the homotopy theory sense, see [8]) between this inductive type and our translation of identity. It is obtained by using the induction principle associated with $\texttt{paths}_\texttt{R}$ in one direction; and by nested path-inductions on p and on the proof of transport along p in the other direction. This indicates that they are morally the same; it may convince the reader that the results of the last two sections could have been carried out using the encoding instead of the primitive notion of identity types.

6.4 Dealing with axioms

While formalizing proofs that need axioms which are independent, it is a common practice to simply add them in the context. Then, if one want to use the parametricity translation, he also needs to provide a witness of the parametricity of the axiom. Therefore axioms that can prove their own parametricity are well-behaved with respect to the translation. More formally, we say that a closed type $P : \mathsf{Type}$ is *provably parametric* if the type $\forall h : P, h \in [\![P]\!]$ is inhabited. We will now give two examples of axioms using identity types which are provably parametric.

- *uniqueness of identity proofs* (UIP) : Let $\texttt{uip}$ be the following type $\texttt{uip} \equiv \forall X : \mathsf{Type}, x\, y : X, p\, q : x = y. p = q$. We want to find an inhabitant of $\forall f : \texttt{uip}. f \in [\![\texttt{uip}]\!]$. The statement $f \in [\![\texttt{uip}]\!]$ unfolds into $\forall [\![X : \mathsf{Type}, x\, y : X, p\, q : x = y]\!].(f\, A\, x\, y\, p\, q)_*(p_R) = q_R$. The conclusion is an equality between paths, so it is provable using f.
- *function extensionality* : Let $\texttt{funext}$ be the following type $\texttt{funext} \equiv \forall A : \mathsf{Type}, B : A \to \mathsf{Type}, f\, g : \forall x : A. Bx. (\forall x : A. f\, x = g\, x) \to f = g$. In his original development, Voedvoesky showed that $\texttt{funext}$ is logically equivalent (there is a function in both directions) to the so-called *weak extensionality* (see [8]). It is defined by $\texttt{weakext} \equiv \forall A : \mathsf{Type}, P : A \to \mathsf{Type}. (\forall x : A. \texttt{Contr}\, P\, x) \to \texttt{Contr}\, (\forall x : A. P\, x))$ where $\texttt{Contr}\, A \equiv \exists x : A. \forall y : A. x = y$ is the predicate for *contractible types*; ie. types which are equivalent to the singleton type. We now sketch the proof that $\texttt{weakext}$ is provably parametric (and thus so is $\texttt{funext}$). The main idea is to notice that $M \in [\![\texttt{Contr}\, A]\!]$ is logically equivalent to $\texttt{Contr}\, ([\![A]\!]\, M_1)$

where M_1 is the first projection of M. The unfolding of $k \in [\![\texttt{weakext}]\!]$ is:

$$\forall A : \mathsf{Type}, A_R : A \to \mathsf{Type}, P : A \to \mathsf{Type}, P_R : (\forall x : A, x_R : A_R\, x.P\, x \to \mathsf{Type}),$$
$$\phi : (\forall x : A.\texttt{Contr}\,(P\, x)), \phi_R : (\forall x : A, x_R : A_R\, x.(\phi\, x) \in [\![\texttt{Contr}\, P\, x]\!]).$$
$$(k\, A\, P\, \phi) \in [\![\texttt{Contr}\,(\forall x : A.P\, x)]\!]$$

Therefore, using the logical equivalence in one direction we can deduce the conclusion from $\texttt{Contr}\,(k\, A\, P\, \phi)_1 \in [\![\forall x : A.P\, x)]\!])$. Then using $k : \texttt{weakext}$ two times, it is enough to prove that $\forall x : A, x_R : A_R\, x.\texttt{Contr}\,((k\, A\, P\, \phi)_1 x) \in [\![P\, x]\!])$ (1). Notice that $(k\, A\, P\, \phi)_1\, x =_A (\phi\, x)_1$ because A is contractible. So we can transport along this path in (1) to obtain $\forall x : A, x_R : A_R\, x.\texttt{Contr}\,((\phi x)_1 \in [\![P\, x]\!])$ (2). Now, using the other direction of logical implication, (2) is implied by $\forall x : A, x_R : A_R\, x.(\phi x) \in [\![\texttt{Contr}\,(P\, x)]\!])$ which is exactly the type of ϕ_R.

7 Conclusion

This work shows that parametricity theory may be used to deduce properties about the algebraic structure of identity types. It allows to give formal arguments to prove canonicity results about definitions in a proof-relevant setting.

The most important question that remains open is whether or not we can extend the translation and the uniqueness property of groupoid laws to deal with Voevodsky's univalence axiom. One way to solve this question would be to prove that univalence axiom is provably parametric which would yield to a positive answer to the question of the compatibility of parametricity theory and univalence. Regardless of the answer to this problem, solving it would give a better understanding of polymorphic type quantifications in univalent universes.

References

[1] Henk Barendregt. Lambda Calculi with Types. In *Handbook of Logic in Computer Science*, pages 117–309. Oxford University Press, 1992.

[2] Jean-Philippe Bernardy, Patrik Jansson, and Ross Paterson. Proofs for free - parametricity for dependent types. *J. Funct. Program.*, 22(2):107–152, 2012.

[3] Guillaume Brunerie. Syntactic grothendieck weak ∞-groupoids. Available as http://uf-ias-2012.wikispaces.com/file/view/SyntacticInfinityGroupoidsRawDefinition.pdf.

[4] Peter Dybjer. Inductive families. *Formal Asp. Comput.*, 6(4):440–465, 1994.

[5] Peter LeFanu Lumsdaine. Weak omega-categories from intensional type theory. In Pierre-Louis Curien, editor, *TLCA*, volume 5608 of *Lecture Notes in Computer Science*, pages 172–187. Springer, 2009.

[6] Zhaohui Luo. Ecc, an extended calculus of constructions. In *LICS*, pages 386–395. IEEE Computer Society, 1989.

[7] John C. Reynolds. Types, Abstraction and Parametric Polymorphism. In *IFIP Congress*, pages 513–523, 1983.

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

[9] Benno van den Berg and Richard Garner. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society*, 102(2):370–394, 2011.