

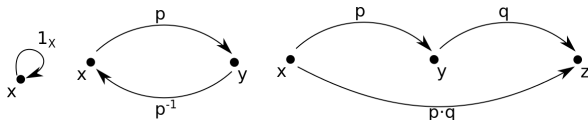
All derivations of groupoid laws are propositionally  
equal

Marc Lasson  
INRIA – equipe projet  $\pi r2$

May 15, 2014

# The weak $\omega$ -groupoid structure of types

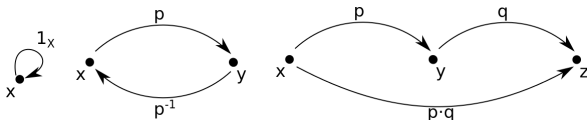
- The groupoid structure of types (Hofmann-Streicher).



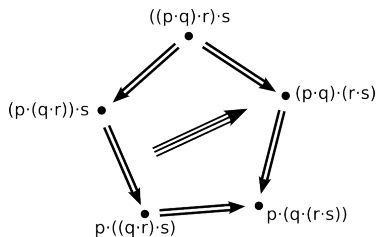
- neutral :  $p \cdot 1 = p$ ,
- assoc :  $p \cdot (q \cdot r) = (p \cdot q) \cdot r$ ,
- involution :  $(p^{-1})^{-1} = p$

# The weak $\omega$ -groupoid structure of types

- The groupoid structure of types (Hofmann-Streicher).



- neutral :  $p \cdot 1 = p$ ,
  - assoc :  $p \cdot (q \cdot r) = (p \cdot q) \cdot r$ ,
  - involution :  $(p^{-1})^{-1} = p$
- Weak  $\omega$ -groupoid structure of types (Garner et al, Lumsdaine).



# The synthetic approach

- Problem: Quite difficult to formalize  $\omega$ -groupoids.
- Synthetic approach : Groupoid laws are proved when needed.

```
Definition left_action {X} {x y : X}
  (p : x = y) {z : X} {q : y = z}
  {r : y = z} (α : q = r)
  : p @ q = p @ r.
induction α; induction q; induction p; reflexivity.
Defined.

Lemma assoc
  {X} {x y z u : X}
  (p : x = y) (q : y = z) (r : z = u) :
  (p @ q) @ r = p @ (q @ r).
induction r; induction q; induction p.
reflexivity.
Defined.

Lemma pentagon :
  forall X (x y z u v : X),
  forall (p : x = y)
    (q : y = z)
    (r : z = u)
    (s : u = v),
  (right_action (assoc p q r) s)
  @ (assoc p (q @ r) s)
  @ (left_action p (assoc q r s))
= (assoc (p @ q) r s) @ (assoc p q (r @ s)).
intros.
induction s; induction r; induction q; induction p.
simpl.
reflexivity.
Defined.
```

Question: Do proof terms matter ?

# Outline

## ① Parametricity with identity types.

### Canonicity of identity functions:

The identity function is the only term inhabiting  $\forall X.X \rightarrow X$ .

## ② Polymorphic loop spaces.

### Canonicity of reflexivities in loop spaces:

The reflexivity is the only term inhabiting a polymorphic loop space.

## ③ Syntactic approach of groupoid laws.

### Canonicity of groupoid laws:

There is only one implementation of a given groupoid law.

# Dependent parametricity theory in a nutshell

Logical predicates:

$$f \in |\forall x : A. B| \equiv \forall x : A, x_R : x \in |A|. (f x) \in |B|$$

Contexts:

$$\llbracket \Gamma, x : A \rrbracket \equiv \llbracket \Gamma \rrbracket, x : A, x_R : x \in |A|$$

Abstraction theorem:

$$\frac{\Gamma \vdash M : A}{\llbracket \Gamma \rrbracket \vdash \llbracket M \rrbracket : M \in |A|}$$

Full definition:

$$\llbracket \lambda x : A. M \rrbracket = \lambda x : A, x_R : x \in |A|. \llbracket M \rrbracket$$

$$\llbracket M N \rrbracket = \llbracket M \rrbracket N \llbracket N \rrbracket$$

$$\llbracket x \rrbracket = x_R$$

$$\llbracket \forall x : A. B \rrbracket = \lambda f : \forall x : A. B. \forall x : A, x_R : x \in |A|. (f x) \in |B|$$

$$\llbracket \text{Type} \rrbracket = \lambda \alpha : \text{Type}. \alpha \rightarrow \text{Type}$$

we have  $M \in |A| \equiv \llbracket A \rrbracket M$ .

# The simplest “Theorem for free”

In standard type theories, the type ID

$$\forall X : \text{Type}. X \rightarrow X$$

is not provably uniquely inhabited. I.e. you cannot prove :

$$\forall f : \text{ID}, X : \text{Type}, x : X. f X x = x$$

But you can prove it is “unique in the syntax”:

# The simplest “Theorem for free”

In standard type theories, the type ID

$$\forall X : \text{Type}. X \rightarrow X$$

is not provably uniquely inhabited. I.e. you cannot prove :

$$\forall f : \text{ID}, X : \text{Type}, x : X. f X x = x$$

But you can prove it is “unique in the syntax”:

## Theorem (Canonicity of the type of the identity)

If  $\vdash M : \text{ID}$ , then there exists :

$$\vdash \pi_M : \forall X x. M X x = x$$

### proof.

The abstraction theorem gives :

$$\vdash \llbracket M \rrbracket : M \in |\forall X : \text{Type}. X \rightarrow X|$$

which unfolds to a proof of :

$$\forall X : \text{Type}, X_R : X \rightarrow \text{Type}, x : X. X_R x \rightarrow X_R (f X x)$$

We conclude by instantiating  $X_R := \lambda y : X. y = x$ .  $\square$



# Identity types and parametricity

Type constructor:

$$\frac{M : A \quad N : A}{M =_A N : \text{Type}}$$

Introduction:

$$\frac{M : A}{1_M : M =_A M}$$

Elimination:

$x : A, p : M = x \vdash P : \text{Type}$

$B : P[M/x, 1_M/p]$

$$\frac{N : A \quad U : M = N}{J(B, N, U) : P[N/x, U/p]}$$

Computation:

$$J(B, M, 1) \equiv B$$

Transport:

$$\frac{U : M = N \quad B : P[M/x]}{U_*(B) : P[N/x]}$$

# Identity types and parametricity

## Translation of identity types:

### Type constructor:

$$\frac{M : A \quad N : A}{M =_A N : \text{Type}}$$

$$U \in |M = N| \quad \equiv \quad U_*(\llbracket M \rrbracket) = \llbracket N \rrbracket$$

### Introduction:

$$\frac{M : A}{1_M : M =_A M}$$

### Translation of reflexivity:

$$\begin{aligned} \llbracket 1_M \rrbracket & : 1_M \in |M = M| \\ & : (1_M)_*(\llbracket M \rrbracket) = \llbracket M \rrbracket \\ & : \llbracket M \rrbracket = \llbracket M \rrbracket \end{aligned}$$

### Elimination:

$$\frac{x : A, p : M = x \vdash P : \text{Type} \quad B : P[M/x, 1_M/p]}{N : A \quad U : M = N \quad \frac{N : A \quad U : M = N}{J(B, N, U) : P[N/x, U/p]}}$$

### Translation of elimination:

$$\llbracket J(B, N, U) \rrbracket = J(J(\llbracket B \rrbracket, N, U), \llbracket N \rrbracket, \llbracket U \rrbracket)$$

### Computation:

$$J(B, M, 1) \equiv B$$

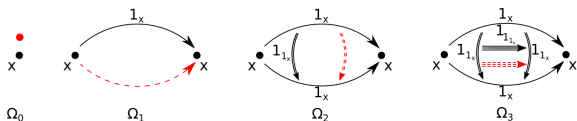
### Translation of computations:

$$\begin{aligned} \llbracket J(B, M, 1) \rrbracket & \equiv J(J(\llbracket B \rrbracket, M, 1), \llbracket M \rrbracket, \llbracket 1 \rrbracket) \\ & \equiv J(\llbracket B \rrbracket, \llbracket M \rrbracket, \llbracket 1 \rrbracket) \\ & \equiv J(\llbracket B \rrbracket, \llbracket M \rrbracket, 1) \\ & \equiv \llbracket B \rrbracket \end{aligned}$$

### Transport:

$$\frac{U : M = N \quad B : P[M/x]}{U_*(B) : P[N/x]}$$

# Loop spaces



$$\Omega_n : \forall X, X \rightarrow \text{Type}$$

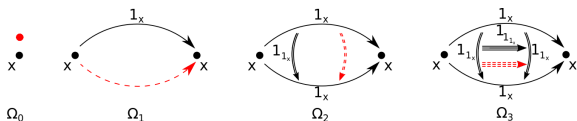
$$\Omega_0(X, x) := X$$

$$\Omega_{n+1}(X, x) := \Omega_n(x = x, 1)$$

$$\omega_n : \forall X, x. \Omega_n(X, x)$$

$$\omega_n(X, x) \equiv 1_{\dots 1_x} \} \quad n \text{ times}$$

# Loop spaces



$$\Omega_n : \forall X, X \rightarrow \text{Type}$$

$$\Omega_0(X, x) := X$$

$$\Omega_{n+1}(X, x) := \Omega_n(x = x, 1)$$

$$\omega_n : \forall X x. \Omega_n(X, x)$$

$$\omega_n(X, x) \equiv 1_{\dots 1_x} \} \text{ } n \text{ times}$$

## Theorem (Canonicity of $\omega_n$ in $\Omega_n$ )

If  $\vdash M : \forall X x. \Omega_n(X, x)$ , then there exists :

$$\vdash \pi_M : \forall X x. M X x = \omega_n(X, x)$$

**proof.** We can prove :

$$p \in |\Omega_n(X, x)|[\lambda y : X. y = x / X_R] \rightarrow p = \omega_n(X, x)$$

by induction over  $n$  and by doing some algebra.

We conclude using  $\llbracket M \rrbracket : \forall X X_R x x_R. (M X x) \in |\Omega_n(X, x)|$ .  $\square$

# Syntactic characterisation of groupoid laws

- Inspired by Guillaume Brunerie's notes.

# Syntactic characterisation of groupoid laws

- Inspired by Guillaume Brunerie's notes.
- **Contractible context:**

$X : \text{Type}, x : X, x_1 : C_1, p_1 : M_1 = x_1, \dots, x_n : C_n, p_n : M_n = x_n$

where  $x_i$  does not occur in  $M_j$ .

# Syntactic characterisation of groupoid laws

- Inspired by Guillaume Brunerie's notes.

- **Contractible context:**

$X : \text{Type}, x : X, x_1 : C_1, p_1 : M_1 = x_1, \dots, x_n : C_n, p_n : M_n = x_n$

where  $x_i$  does not occur in  $M_i$ .

- Let **MLID** be a minimal fragment of type theory, with:
  - Identity types (intro, elim, computation),
  - and restricted to contractible contexts.

No function spaces, universes, sigma types, nor inductive families.

# Syntactic characterisation of groupoid laws

- Inspired by Guillaume Brunerie's notes.

- **Contractible context:**

$X : \text{Type}, x : X, x_1 : C_1, p_1 : M_1 = x_1, \dots, x_n : C_n, p_n : M_n = x_n$

where  $x_i$  does not occur in  $M_i$ .

- Let **MLID** be a minimal fragment of type theory, with:
  - Identity types (intro, elim, computation),
  - and restricted to contractible contexts.

No function spaces, universes, sigma types, nor inductive families.

- A **groupoid law** is a type  $\forall \Gamma. C$  such that :

$$\Gamma \vdash_{\text{Id}} C : \text{Type}$$

with  $\Gamma$  a contractible context.



# Syntactic characterisation of groupoid laws

- Inspired by Guillaume Brunerie's notes.

- **Contractible context:**

$X : \text{Type}, x : X, x_1 : C_1, p_1 : M_1 = x_1, \dots, x_n : C_n, p_n : M_n = x_n$

where  $x_i$  does not occur in  $M_i$ .

- Let **MLID** be a minimal fragment of type theory, with:
  - Identity types (intro, elim, computation),
  - and restricted to contractible contexts.

No function spaces, universes, sigma types, nor inductive families.

- A **groupoid law** is a type  $\forall \Gamma. C$  such that :

$$\Gamma \vdash_{\text{id}} C : \text{Type}$$

with  $\Gamma$  a contractible context.

**Lemma (Groupoid laws in " $X : \text{Type}, x : X$ " are loop spaces)**

*If  $X : \text{Type}, x : X \vdash_{\text{id}} M : C$ , there exists  $n$  such that*

$$M \equiv \omega_n(X, x) \quad C \equiv \Omega_n(X, x)$$

## Theorem (Groupoid laws are inhabited)

If  $\Gamma \vdash_{\text{id}} C : \text{Type}$  is a groupoid law, there exists  $\Gamma \vdash_{\text{id}} \Theta_{\Gamma.C} : C$ .

Example:

```
pentagon.v
Lemma pentagon X (x : X)
  y (p : x = y)
  z (q : y = z)
  u (r : z = u)
  v (s : u = v) :
  (right_action (assoc p q r) s)
  @ (assoc p (q @ r) s)
  @ (left_action p (assoc q r s))
  = (assoc (p @ q) r s) @ (assoc p q (r @ s)).
induction s.
induction r.
induction q.
induction p.
simpl.
reflexivity.
Defined.
```

```
1 subgoal
X : Type
x : X
y : X
p : x = y
z : X
q : y = z
u : X
r : z = u
v : X
s : u = v
(1/1)
(right_action (assoc p q r) s @
  assoc p (q @ r) s) @
left_action p (assoc q r s) =
assoc (p @ q) r s @ assoc p q (r @ s)
```

Ready, proving pentagon

Line: 50 Char: 9 CoqIDE started

## Theorem (Groupoid laws are inhabited)

If  $\Gamma \vdash_{\text{id}} C : \text{Type}$  is a groupoid law, there exists  $\Gamma \vdash_{\text{id}} \Theta_{\Gamma, C} : C$ .

Example:

The screenshot shows a proof assistant interface with a toolbar at the top containing icons for file operations, undo, redo, search, and help. The main window is titled 'pentagon.v' and is split into two panes. The left pane contains the following code:

```
Lemma pentagon X (x : X)
  y (p : x = y)
  z (q : y = z)
  u (r : z = u)
  v (s : u = v) :
  (right_action (assoc p q r) s)
  @ (assoc p (q @ r) s)
  @ (left_action p (assoc q r s))
  = (assoc (p @ q) r s) @ (assoc p q (r @ s)).
induction s.
induction r.
induction q.
induction p.
simpl.
reflexivity.
Defined.
```

The right pane shows the current subgoal:

```
1 subgoal
X : Type
x : X
y : X
p : x = y
z : X
q : y = z
u : X
r : z = u
_____ (1/1)
(right_action (assoc p q r) 1 @
  assoc p (q @ r) 1) @
left_action p (assoc q r 1) =
assoc (p @ q) r 1 @ assoc p q (r @ 1)
```

At the bottom of the interface, a status bar indicates 'Ready, proving pentagon', 'Line: 44 Char: 13', and a 'CoqIDE started' button.

## Theorem (Groupoid laws are inhabited)

If  $\Gamma \vdash_{\text{id}} C : \text{Type}$  is a groupoid law, there exists  $\Gamma \vdash_{\text{id}} \Theta_{\Gamma.C} : C$ .

Example:

The screenshot shows a proof assistant interface with a toolbar at the top containing icons for file operations, navigation, and help. The main window is titled "pentagon.v" and is split into two panes. The left pane contains the source code for a lemma, and the right pane shows the current subgoal.

```
Lemma pentagon X (x : X)
  y (p : x = y)
  z (q : y = z)
  u (r : z = u)
  v (s : u = v) :
  (right_action (assoc p q r) s)
  @ (assoc p (q @ r) s)
  @ (left_action p (assoc q r s))
  = (assoc (p @ q) r s) @ (assoc p q (r @ s)).
induction s.
induction r.
induction q.
induction p.
simpl.
reflexivity.
Defined.
```

The right pane shows the subgoal:

```
1 subgoal
X : Type
x : X
y : X
p : x = y
z : X
q : y = z
_____ (1/1)
(right_action (assoc p q 1) 1 @
  assoc p (q @ 1) 1) @
left_action p (assoc q 1 1) =
assoc (p @ q) 1 1 @ assoc p q (1 @ 1)
```

At the bottom of the interface, a status bar indicates "Ready, proving pentagon" on the left, "Line: 45 Char: 13" in the center, and "CoqIDE started" on the right.

## Theorem (Groupoid laws are inhabited)

If  $\Gamma \vdash_{\text{id}} C : \text{Type}$  is a groupoid law, there exists  $\Gamma \vdash_{\text{id}} \Theta_{\Gamma.C} : C$ .

Example:

The screenshot shows a Coq IDE window titled "pentagon.v". The left pane contains the following code:

```
Lemma pentagon X (x : X)
  y (p : x = y)
  z (q : y = z)
  u (r : z = u)
  v (s : u = v) :
  (right_action (assoc p q r) s)
  @ (assoc p (q @ r) s)
  @ (left_action p (assoc q r s))
  = (assoc (p @ q) r s) @ (assoc p q (r @ s)).
induction s.
induction r.
induction q.
induction p.
simpl.
reflexivity.
Defined.
```

The right pane shows a subgoal:

```
1 subgoal
X : Type
x : X
y : X
p : x = y
----- (1/1)
(right_action (assoc p 1 1) 1 @
  assoc p (1 @ 1) 1) @
left_action p (assoc 1 1 1) =
assoc (p @ 1) 1 1 @ assoc p 1 (1 @ 1)
```

At the bottom of the IDE, the status bar reads "Ready, proving pentagon" and "Line: 46 Char: 13 CoqIde started".

## Theorem (Groupoid laws are inhabited)

If  $\Gamma \vdash_{\text{id}} C : \text{Type}$  is a groupoid law, there exists  $\Gamma \vdash_{\text{id}} \Theta_{\Gamma, C} : C$ .

Example:

The screenshot shows a Coq IDE window titled "pentagon.v". The left pane contains the following code:

```
Lemma pentagon X (x : X)
  y (p : x = y)
  z (q : y = z)
  u (r : z = u)
  v (s : u = v) :
  (right_action (assoc p q r) s)
  @ (assoc p (q @ r) s)
  @ (left_action p (assoc q r s))
  = (assoc (p @ q) r s) @ (assoc p q (r @ s)).
induction s.
induction r.
induction q.
induction p.
simpl.
reflexivity.
Defined.
```

The right pane shows a subgoal:

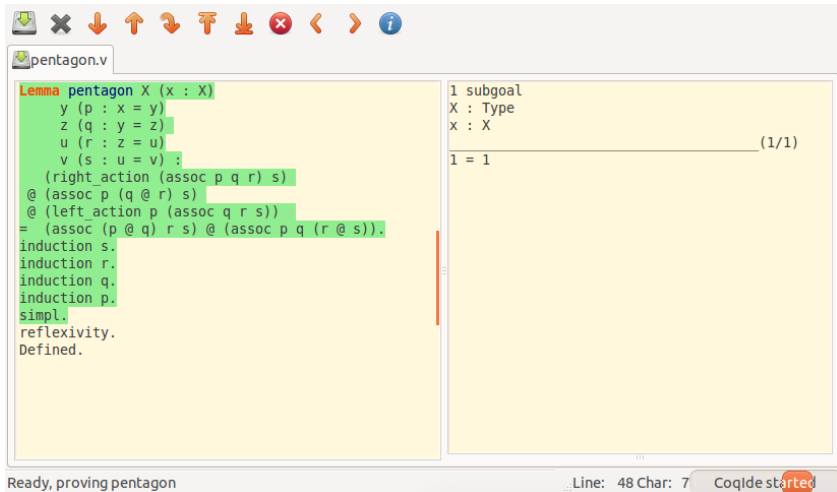
```
1 subgoal
X : Type
x : X
----- (1/1)
(right_action (assoc 1 1 1) 1 @
  assoc 1 (1 @ 1) 1) @
left_action 1 (assoc 1 1 1) =
assoc (1 @ 1) 1 1 @ assoc 1 1 (1 @ 1)
```

At the bottom of the IDE, the status bar reads "Ready, proving pentagon" and "Line: 47 Char: 13 CoqIde started".

## Theorem (Groupoid laws are inhabited)

If  $\Gamma \vdash_{\text{id}} C : \text{Type}$  is a groupoid law, there exists  $\Gamma \vdash_{\text{id}} \Theta_{\Gamma.C} : C$ .

Example:



The screenshot shows a Coq IDE window titled "pentagon.v". The left pane contains the following code:

```
Lemma pentagon X (x : X)
  y (p : x = y)
  z (q : y = z)
  u (r : z = u)
  v (s : u = v) :
  (right_action (assoc p q r) s)
  @ (assoc p (q @ r) s)
  @ (left_action p (assoc q r s))
  = (assoc (p @ q) r s) @ (assoc p q (r @ s)).
induction s.
induction r.
induction q.
induction p.
simpl.
reflexivity.
Defined.
```

The right pane shows a subgoal:

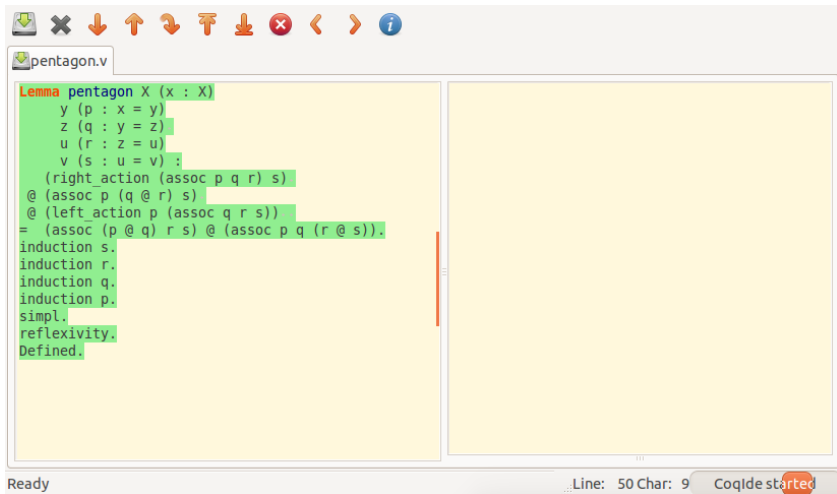
```
1 subgoal
X : Type
x : X
----- (1/1)
1 = 1
```

At the bottom of the IDE, the status bar indicates "Ready, proving pentagon" and "Line: 48 Char: 7 CoqIde started".

## Theorem (Groupoid laws are inhabited)

If  $\Gamma \vdash_{\text{id}} C : \text{Type}$  is a groupoid law, there exists  $\Gamma \vdash_{\text{id}} \Theta_{\Gamma.C} : C$ .

Example:



The screenshot shows a Coq IDE window titled "pentagon.v". The editor contains the following code:

```
Lemma pentagon X (x : X)
  y (p : x = y)
  z (q : y = z)
  u (r : z = u)
  v (s : u = v) :
  (right_action (assoc p q r) s)
  @ (assoc p (q @ r) s)
  @ (left_action p (assoc q r s))
  = (assoc (p @ q) r s) @ (assoc p q (r @ s)).
induction s.
induction r.
induction q.
induction p.
simpl.
reflexivity.
Defined.
```

The status bar at the bottom indicates "Ready", "Line: 50 Char: 9", and "CoqIde started".



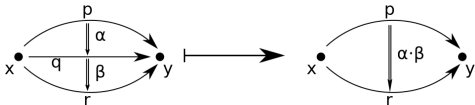
# Unicity of the canonical inhabitant

## Theorem (Canonicity of proofs of groupoid laws)

If  $\forall \Gamma. C$  is a groupoid law and  $\vdash M : \forall \Gamma. C$  and  $\vdash M' : \forall \Gamma. C$  then there exists  $\vdash \pi_{M, M'} : \forall \vec{\gamma} : \Gamma. (M \vec{\gamma}) = (M' \vec{\gamma})$

Example **vertical composition**:

**forall**  $X (x\ y : X) (p : x = y) (q : x = y) (\alpha : p = q)$   
 $(z : X) (\beta : q = r), p = r.$



We want to prove that

**forall**  $X\ x\ y\ p\ q\ \alpha\ z\ \beta, M\ X\ x\ y\ p\ q\ \alpha\ z\ \beta = M'\ X\ x\ y\ p\ q\ \alpha\ z\ \beta$

By successive inductions, it is enough to prove that :

**forall**  $X\ x, M\ X\ x\ x\ 1\ 1\ 1\ x\ 1 = M'\ X\ x\ x\ 1\ 1\ 1\ x\ 1$

But LHS and RHS are an inhabitant of  $\Omega_2(X, x)$  !

Using the canonicity for loop spaces, they are equal to  $\omega_2(X, x)$ .

# Conclusion

Open problems :

- Comparing models of MLID with definitions of groupoids.
- Compatibility with axioms:

UIP / K	✓
Proof-irrelevance	✓
Extensionality	✓
Excluded middle	✗
Univalence	???