Computing with Univalence

Daniel R. Licata
with Robert Harper

Carnegie Mellon University

Identity Types

$$\frac{A \text{ type} \qquad M, N : A}{\mathrm{Id}_A(M,N) \text{ type}}$$

Identity Types

$$\frac{A \text{ type} \qquad M, N : A}{\mathrm{Id}_A(M,N) \text{ type}}$$

$$\frac{C : A \to \text{type} \qquad \alpha : \mathrm{Id}_A(M,N) \qquad P : C[M]}{\mathrm{subst}_C\ \alpha\ P : C[N]}$$

$$\frac{}{\mathrm{Refl} : \mathrm{Id}_A(M,M)}$$

$$\frac{F : A \to B \qquad \alpha : \mathrm{Id}_A(M,N)}{\mathrm{resp}\ F\ \alpha : \mathrm{Id}_B(F\ M, F\ N)}$$

all families respect identity

Identity Types

$$\frac{A \text{ type} \qquad M, N : A}{Id_A(M,N) \text{ type}}$$

$$\frac{C : A \to \text{type} \qquad \alpha : Id_A(M,N) \qquad P : C[M]}{subst_C\ \alpha\ P : C[N]}$$

$$\frac{}{Refl : Id_A(M,M)}$$

$$\frac{F : A \to B \qquad \alpha : Id_A(M,N)}{resp\ F\ \alpha : Id_B(F\ M, F\ N)}$$

all families respect identity

Computation rules: $subst_C\ Refl\ P \equiv P$

$resp\ F\ Refl \equiv Refl$

definitional equality: used silently

Id in HoTT = Paths

$$\frac{}{\mathsf{Refl} : \mathsf{Id}_A(M,M)} \qquad \frac{C : A \to \mathsf{type} \quad \alpha : \mathsf{Id}_A(M,N) \quad P : C[M]}{\mathsf{subst}_C\ \alpha\ P : C[N]} \qquad \frac{F : A \to B \quad \alpha : \mathsf{Id}_A(M,N)}{\mathsf{resp}\ F\ \alpha : \mathsf{Id}_B(F\ M, F\ N)}$$

- **Not** the least type containing reflexivity
  (we'll add other canonical inhabitants soon)
- **Not** the greatest type everything respects
  (not forced to include all consistent equalities)

Id in HoTT = Paths

To define a type, you get to specify:

- what are its members? $M : A$

- what are its paths? $\alpha : Id_A(M,N)$
  (really a judgement $\alpha : M \simeq_A N$)

Refl subst resp (and a few more operations) are an **interface** that you must implement

Id in HoTT = Paths

To define a type, you get to specify:

- what are its members? $M : A$

- what are its paths? $\alpha : Id_A(M,N)$
  (really a judgement $\alpha : M \simeq_A N$)

Refl subst resp (and a few more operations) are an **interface** that you must implement

generalization of the Martin-Löf/NuPRL methodology of members and **equality…**

Univalence

[Hofmann&Streicher, Voevodsky]

“Isomorphic types are identical”

Univalence

[Hofmann&Streicher, Voevodsky]

“Isomorphic types are identical”

in the sense of Id: interchangeable in all contexts

Univalence

[Hofmann&Streicher, Voevodsky]

“Isomorphic types are identical”

really means homotopy equivalence

in the sense of Id: interchangeable in all contexts

Univalence

A, B : U ⟵ **universe = type of (small) types**

Univalence axiom entails the existence of a term

$ua : HEquiv(A,B) \to Id_U(A,B)$

$$\frac{f : A \to B \qquad \alpha : Id_{A\to A}(g \circ f, id) \qquad g : B \to A \qquad \beta : Id_{B\to B}(f \circ g, id)}{(f,g,\alpha,\beta) : HEquiv(A,B)}$$

Univalence

A, B : U ⟵ **universe = type of (small) types**

Univalence axiom entails the existence of a term

$ua : HEquiv(A,B) \to Id_U(A,B)$

$$\frac{f : A \to B \qquad \alpha : Id_{A\to A}(g \circ f, id) \qquad g : B \to A \qquad \beta : Id_{B\to B}(f \circ g, id)}{(f,g,\alpha,\beta) : HEquiv(A,B)}$$

HEquiv, and therefore Id, have computational content!
They are *structures, not propositions*

“Isomorphic Types are Interchangeable”

$$ua : HEquiv(A,B) \to Id_U(A,B)$$

$$\frac{\begin{array}{l} C : U \to U \\ (f,g,\alpha,\beta) : HEquiv(A,B) \\ P : C[A] \end{array}}{subst_C\ ua(f,g,\alpha,\beta)\ P : C[B]}$$

Constructivity

Standard computation rule

$$\mathrm{subst}_C\ \mathrm{Refl}\ P \equiv P$$

is clearly insufficient:

$$\mathrm{subst}_C\ \mathrm{ua}(f,g,\alpha,\beta)\ P$$

is *stuck*!

added an introduction rule without adding a new β rule

Constructivity

Standard computation rule

$$\mathrm{subst}_C\ \mathrm{Refl}\ P \equiv P$$

is clearly insufficient:

$$\mathrm{subst}_C\ \mathrm{ua}(f,g,\alpha,\beta)\ P$$

is *stuck*!

added an introduction rule without adding a new β rule

Open problem: Can you normalize a program that is written using univalence?

Constructivity

Open problem: Can you normalize a program that is written using univalence?

Conjecture: yes!

- univalence is admissible in type theory **without** U: all constructions respect type isomorphism
- UA demands that constructions on U do too
- should be able to eliminate UA using admissibility

Approaches to Computation

- Constructive model in simplicial sets/types [Coquand&Huber, next]
- or in weak ω-groupoids [Altenkirch et al., later today]
- **This work:** progress on special case using syntactic techniques inspired by groupoid model

Canonicity for 2D type theory

[Licata&Harper, POPL'12]

- Take various equations from the Hofmann/Streicher groupoid model as (strict) definitional equalities ≡
- Consistency by groupoid model
- Prove **canonicity**

  If $\cdot \vdash M : 2$ then $M \equiv \mathsf{true}$ or $M \equiv \mathsf{false}$

by "logical groupoids"

Canonicity for 2D type theory

If $\cdot \vdash M : 2$ then $M \equiv \mathsf{true}$ or $M \equiv \mathsf{false}$

Simplifications:

- 2 dimensional means
  - E.g. $\mathrm{Id}_U(A,B)$ can have computational content…
  - but $\mathrm{Id}_{\mathrm{Id}(M,N)}(\alpha,\beta)$ is trivial (equality reflection + UIP).

Canonicity for 2D type theory

If $\cdot \vdash M : 2$ then $M \equiv \mathsf{true}$ or $M \equiv \mathsf{false}$

Simplifications:

- 2 dimensional means
  - E.g. $\mathrm{Id}_U(A,B)$ can have computational content…
  - but $\mathrm{Id}_{\mathrm{Id}(M,N)}(\alpha,\beta)$ is trivial (equality reflection + UIP). All paths between paths are reflexivity:

$$\frac{\beta : \mathrm{Id}_{\mathrm{Id}(M,N)}(\alpha_1,\alpha_2)}{\alpha_1 \equiv \alpha_2} \qquad \frac{\beta : \mathrm{Id}_{\mathrm{Id}(M,N)}(\alpha_1,\alpha_2)}{\beta \equiv \mathrm{Refl}}$$

Canonicity for 2D type theory

If $\cdot \vdash M : 2$ then $M \equiv \mathsf{true}$ or $M \equiv \mathsf{false}$

Simplifications:

- 2 dimensional means
  - E.g. $\mathrm{Id}_U(A,B)$ can have computational content…
  - but $\mathrm{Id}_{\mathrm{Id}(M,N)}(\alpha,\beta)$ is trivial (equality reflection + UIP). We avoid normalizing homotopies.
- Only proved canonicity up to $\equiv$, which includes equality reflection: not yet an algorithm.

Canonicity for 2D type theory

If $\cdot \vdash M : 2$ then $M \equiv \mathsf{true}$ or $M \equiv \mathsf{false}$

Simplifications:

- 2 dimensional means
  - E.g. $\mathrm{Id}_U(A,B)$ can have computational content…
  - but $\mathrm{Id}_{\mathrm{Id}(M,N)}(\alpha,\beta)$ is trivial (equality reflection + UIP). We avoid normalizing homotopies.
- Only proved canonicity up to $\equiv$, which includes equality reflection: not yet an algorithm.

But proof suggests an algorithm for 2D, and perhaps for low levels of ∞D

Key Idea

$$\frac{C : A \to \mathrm{type} \qquad \alpha : \mathrm{Id}_A(M,N) \qquad P : C[M]}{\mathrm{subst}_C\ \alpha\ P : C[N]} \qquad \frac{F : A \to B \qquad \alpha : \mathrm{Id}_A(M,N)}{\mathrm{resp}\ F\ \alpha : \mathrm{Id}_B(F\ M, F\ N)}$$

Admissibility of univalence = *generic programs:*

- subst computes guided by the structure of C
- resp computes guided by the structure of F
- similar non-uniform definitions for other groupoid operations

not surprising from semantics
but need to express it in syntax

Key Idea

[Altenkirch et al.'s OTT does this for sets; we do it for groupoids]

$$\frac{C : A \to \mathsf{type} \quad \alpha : \mathsf{Id}_A(M,N) \quad P : C[M]}{\mathsf{subst}_C\ \alpha\ P : C[N]} \qquad \frac{F : A \to B \quad \alpha : \mathsf{Id}_A(M,N)}{\mathsf{resp}\ F\ \alpha : \mathsf{Id}_B(F\ M, F\ N)}$$

Admissibility of univalence = *generic programs:*

- subst computes guided by the structure of C
- resp computes guided by the structure of F
- similar non-uniform definitions for other groupoid operations

not surprising from semantics
but need to express it in syntax

Computational interpretation

1. Define each type by members and paths,
   with Refl $^{-1}$ o satisfying groupoid laws

2. Define $\mathrm{subst}_C\ \alpha\ M$ for each C

3. Define resp F α for each F and α

4. Define full Id-elim rule J using subst

(1,2,3 simultaneous)

Computational interpretation

1. **Define each type by members and paths, with Refl $^{-1}$ o satisfying groupoid laws**

2. Define subst_C α M for each C

3. Define resp F α for each F and α

4. Define full Id-elim rule J using subst

Type A * B

- Members: (M, N) where M : A and N : B
- Paths: $$\frac{\alpha : Id_A(M,N) \quad \beta : Id_B(M', N')}{pair{\simeq}\ \alpha\ \beta : Id_{A*B}(M,N)(M',N')}$$
- $Refl_{(M,N)} \equiv pair{\simeq}\ Refl_M\ Refl_N$
- $(pair{\simeq}\ \alpha\ \beta)^{-1} \equiv (pair{\simeq}\ \alpha^{-1}\ \beta^{-1})$
- $(pair{\simeq}\ \alpha\ \beta)\ o\ (pair{\simeq}\ \alpha'\ \beta') \equiv (pair{\simeq}\ \alpha o \alpha'\ \beta o \beta')$

Type A * B

- Members: (M, N) where M : A and N : B
- Paths: $$\frac{\alpha : Id_A(M,N) \quad \beta : Id_B(M', N')}{pair{\simeq}\ \alpha\ \beta : Id_{A*B}(M,N)(M',N')}$$
- $Refl_{(M,N)} \equiv pair{\simeq}\ Refl_M\ Refl_N$ **justifies Refl rule**
- $(pair{\simeq}\ \alpha\ \beta)^{-1} \equiv (pair{\simeq}\ \alpha^{-1}\ \beta^{-1})$
- $(pair{\simeq}\ \alpha\ \beta)\ o\ (pair{\simeq}\ \alpha'\ \beta') \equiv (pair{\simeq}\ \alpha o \alpha'\ \beta o \beta')$

Type A * B

- Members: (M, N) where M : A and N : B
- Paths: $$\frac{\alpha : Id_A(M,N) \quad \beta : Id_B(M', N')}{pair{\simeq}\ \alpha\ \beta : Id_{A*B}(M,N)(M',N')}$$
- $Refl_{(M,N)} \equiv pair{\simeq}\ Refl_M\ Refl_N$ **justifies Refl rule**
- $(pair{\simeq}\ \alpha\ \beta)^{-1} \equiv (pair{\simeq}\ \alpha^{-1}\ \beta^{-1})$ **needed later**
- $(pair{\simeq}\ \alpha\ \beta)\ o\ (pair{\simeq}\ \alpha'\ \beta') \equiv (pair{\simeq}\ \alpha o \alpha'\ \beta o \beta')$

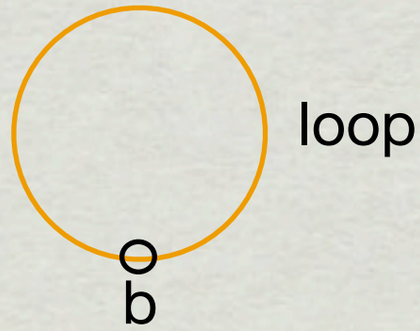
Type U

- Members: <names for small sets>, which determine types by the type El(A : U)
- Paths:

$$\frac{(f,g,\alpha,\beta) : \mathrm{HEquiv}\ A\ B}{\mathrm{ua}(f,g,\alpha,\beta) : \mathrm{Id}_U(A, B)}$$

- $\mathrm{Refl}_A \equiv (x{:}\mathrm{El}(A).x, x{:}\mathrm{El}(A).x, x.\mathrm{Refl}_x, x.\mathrm{Refl}_x)$
- $(\mathrm{ua}(f,g,\alpha,\beta))^{-1} \equiv \ldots$
- $(\mathrm{ua}(f',g',\alpha',\beta'))\ \mathrm{o}\ (\mathrm{ua}(f',g',\alpha',\beta')) \equiv \ldots$

Circle S^1



- Members: b : S^1
- Paths: freely generated by $-^{-1}$ and Refl_b and (- o -)
  from loop : b ≃ b
- such that groupoid laws hold
  up to ≡ for 2TT
  up to ≃ for full HoTT

Computational interpretation

1. Define each type by members and paths, with Refl $^{-1}$ and o satisfying groupoid laws

2. Define $\text{subst}_C\ \alpha\ M$ for each family C

3. Define resp F α for each F and α

4. Define full Id-elim rule J using subst

Subst

$$\frac{C : A \to \text{type} \quad \alpha : \mathrm{Id}_A(M,N) \quad P : C[M]}{\mathrm{subst}_C\ \alpha\ P : C[N]}$$

To be a family of types over A, C must have an associated operation subst_C

Functionality (à la NuPRL) becomes functoriality

Case for x.A(x) * B(x)

Want $\text{subst}_{x.A(x)\ x\ B(x)}\ \alpha$:

Id(θ1,θ2)

$$A[\theta 1] * B[\theta 1] \quad \rightarrow \quad A[\theta 2] * B[\theta 2]$$

Have:

Case for x.A(x) * B(x)

Want $\text{subst}_{\text{x.A(x) x B(x)}}\ \alpha$:

Id(θ1,θ2)

$$A[\theta 1] * B[\theta 1] \quad \rightarrow \quad A[\theta 2] * B[\theta 2]$$

Have:

$$\text{subst}_{A}\ \alpha : A[\theta 1] \rightarrow A[\theta 2]$$

Case for x.A(x) * B(x)

Want $\text{subst}_{x.A(x)\ x\ B(x)}\ \alpha$:

Id(θ1,θ2)

$$A[\theta 1] * B[\theta 1] \quad \rightarrow \quad A[\theta 2] * B[\theta 2]$$

Have:

$$\text{subst}_A\ \alpha : A[\theta 1] \rightarrow A[\theta 2]$$

$$\text{subst}_B\ \alpha : B[\theta 1] \rightarrow B[\theta 2]$$

Case for x.A(x) * B(x)

Id(θ1,θ2)

Want $\text{subst}_{\text{x.A(x) x B(x)}}\ \alpha$:

$$A[\theta 1] * B[\theta 1] \quad \rightarrow \quad A[\theta 2] * B[\theta 2]$$

Have:

$$\text{subst}_{A}\ \alpha : A[\theta 1] \rightarrow A[\theta 2]$$

$$\text{subst}_{B}\ \alpha : B[\theta 1] \rightarrow B[\theta 2]$$

$$\text{subst}_{\text{x.A(x) x B(x)}}\ \alpha\ (M1,M2)$$
$$\equiv (\text{subst}_{\text{x.A(x)}}\ \alpha\ M1,\ \text{subst}_{\text{X:U.B(x)}}\ \alpha\ M2)$$

Case for x.A(x) → B(x)

Id(θ1,θ2)

Want $\text{subst}_{\text{x.A(x)} \to \text{B(x)}}\ \alpha\ F$:

$$A[\theta 2] \quad \to \quad B[\theta 2]$$

$$A[\theta 1] \quad \to \quad B[\theta 1]$$
$$F$$

$$\text{subst}_{\text{x.A(X)} \to \text{B(X)}}\ \alpha\ F$$
$$\equiv (\lambda x.\text{subst}_{\text{xB(X)}}\ \alpha\ (F\ (\text{subst}_{\text{X:U.A(X)}}\ \alpha^{-1}\ x))$$

Case for x.A(x) → B(x)

Id(θ1,θ2)

Want $\text{subst}_{\mathbf{x.A(x) \to B(x)}}\ \alpha\ F$:

$A[\theta 2] \quad \to \quad B[\theta 2]$

$\text{subst}_A\ \alpha^{-1}$

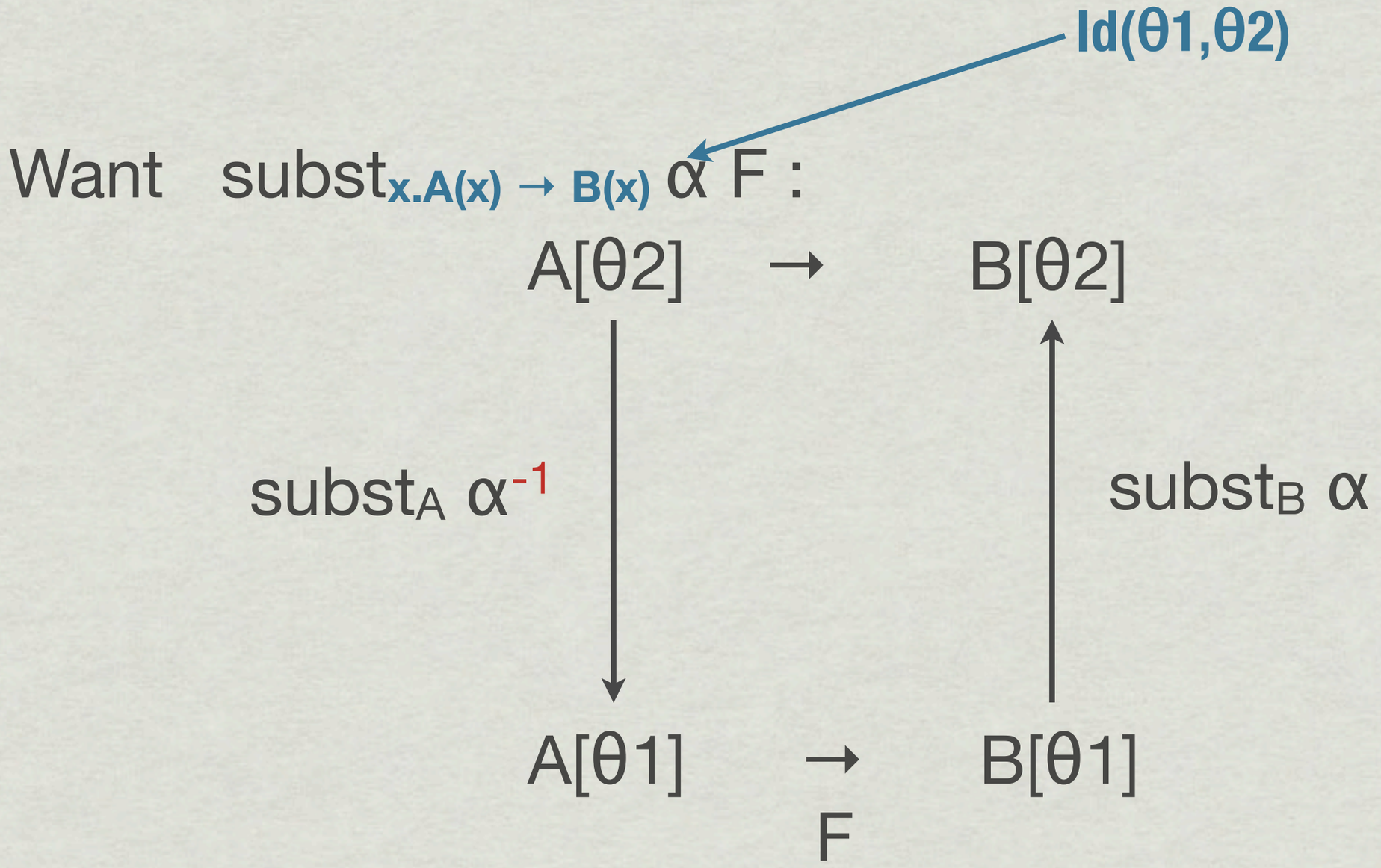
$A[\theta 1] \quad \to \quad B[\theta 1]$

F

$\text{subst}_{\mathbf{x.A(X) \to B(X)}}\ \alpha\ F$

$\equiv (\lambda x.\text{subst}_{\mathbf{xB(X)}}\ \alpha\ (F\ (\text{subst}_{\mathbf{X:U.A(X)}}\ \alpha^{-1}\ x))$

Case for x.A(x) → B(x)

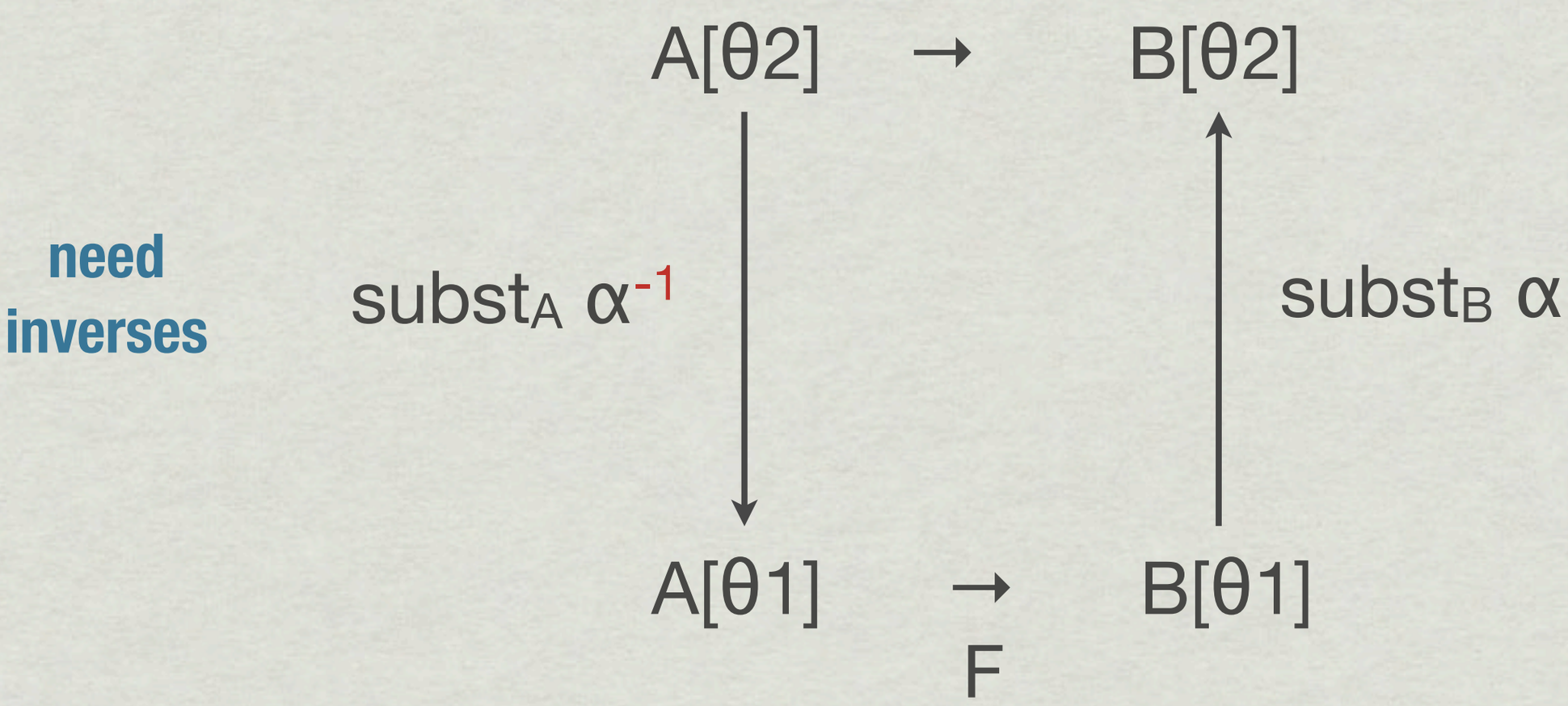


$$\text{subst}_{x.A(X) \to B(X)}\ \alpha\ F$$
$$\equiv (\lambda x.\text{subst}_{xB(X)}\ \alpha\ (F\ (\text{subst}_{X:U.A(X)}\ \alpha^{-1}\ x))$$

Case for x.A(x) → B(x)

Id(θ1,θ2)

Want $\text{subst}_{\text{x.A(x)} \to \text{B(x)}}\ \alpha\ F$:



$\text{subst}_{\text{x.A(X)} \to \text{B(X)}}\ \alpha\ F$

$\equiv (\lambda x.\text{subst}_{\text{xB(X)}}\ \alpha\ (F\ (\text{subst}_{\text{X:U.A(X)}}\ \alpha^{-1}\ x))$

Case for x:U.El(x)

Want $\mathrm{subst}_{\mathbf{x:U.El(x)}}\ \alpha : \mathrm{El}(A) \to \mathrm{El}(B)$

Have $\alpha : \mathrm{Id}_U(A, B)$

Case for x:U.El(x)

Want $\mathrm{subst}_{\mathbf{x:U.El(x)}}\ \alpha : \mathrm{El}(A) \to \mathrm{El}(B)$

Have $\alpha : \mathrm{Id}_U(A, B)$

the only paths in the universe are constructed by univalence

Case for x:U.El(x)

Want $\mathrm{subst}_{\mathbf{x:U.El(x)}}\ \alpha : \mathrm{El}(A) \to \mathrm{El}(B)$

Have $\alpha : \mathrm{Id}_U(A, B)$

the only paths in the universe are constructed by univalence

$\mathrm{subst}_{\mathbf{x.El(x)}}\ \alpha \equiv f$ when $\alpha \equiv \mathrm{ua}(f{:}\mathrm{El}(A)\to\mathrm{El}(B), g, \alpha, \beta)$

Case for x:U.El(x)

Want $\mathrm{subst}_{\mathbf{x:U.El(x)}}\ \alpha : \mathrm{El}(A) \to \mathrm{El}(B)$

Have $\alpha : \mathrm{Id}_U(A, B)$

the only paths in the universe are constructed by univalence

$\mathrm{subst}_{\mathbf{x.El(x)}}\ \alpha \equiv f$ when $\alpha \equiv \mathrm{ua}(f{:}\mathrm{El}(A)\to\mathrm{El}(B), g, \alpha, \beta)$

β-reduction for univalence: deploy the isomorphism

Case for $x.Id_A(M,N)$

Id(θ1,θ2)

Want $subst_{x.IdA(M(x),N(x))}\ \alpha\ \beta$:

$$M[\theta 2] \simeq N[\theta 2]$$

$$M[\theta 1] \simeq N[\theta 1]$$
$$\beta$$

Case for $x.Id_A(M,N)$

Want $subst_{x.IdA(M(x),N(x))}\ \alpha\ \beta$:

Id(θ1,θ2)

$$M[\theta 2] \simeq N[\theta 2]$$

$(resp\ M\ \alpha)^{-1}$ ≀I

$$M[\theta 1] \simeq N[\theta 1]$$

β

Case for $x.\mathrm{Id}_A(M,N)$

Want $\mathrm{subst}_{x.\mathrm{Id}_A(M(x),N(x))}\ \alpha\ \beta$:

Id(θ1,θ2)

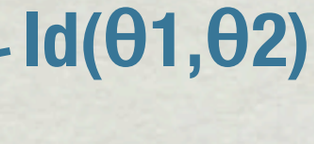
$$M[\theta 2] \simeq N[\theta 2]$$

$$(\mathrm{resp}\ M\ \alpha)^{-1} \wr \qquad \wr\ \mathrm{resp}\ N\ \alpha$$

$$M[\theta 1] \simeq N[\theta 1]$$

$$\beta$$

Case for $x.Id_A(M,N)$



Want $subst_{x.IdA(M(x),N(x))}\ \alpha\ \beta$:

$$M[\theta 2] \simeq N[\theta 2]$$

$$(resp\ M\ \alpha)^{-1} \wr\mid \qquad \mid\wr\ resp\ N\ \alpha$$

$$M[\theta 1] \simeq N[\theta 1]$$

$$\beta$$

$subst_{x.\ IdA(M(x),N(x))}\ \alpha\ \beta$

$\equiv (resp\ N\ \alpha)\ \circ\ \beta\ \circ\ (resp\ M\ \alpha)^{-1}$

Case for $x.Id_A(M,N)$

Id(θ1,θ2)

Want $subst_{x.IdA(M(x),N(x))}\ \alpha\ \beta$:

$$M[\theta 2] \simeq N[\theta 2]$$

$$(resp\ M\ \alpha)^{-1} \wr\mid \qquad \mid\wr\ resp\ N\ \alpha$$

$$M[\theta 1] \simeq N[\theta 1]$$

$$\beta$$

$subst_{x.\ IdA(M(x),N(x))}\ \alpha\ \beta$

$\equiv (resp\ N\ \alpha)\ \circ\ \beta\ \circ\ (resp\ M\ \alpha)^{-1}$

need inverses, composition, resp

action of Hom functor

Case for $x.Id_A(M,N)$

Id(θ1,θ2)

Want $subst_{x.IdA(M(x),N(x))}\ \alpha\ \beta$:

$$M[\theta 2] \simeq N[\theta 2]$$

$$(resp\ M\ \alpha)^{-1} \wr\mid \qquad \mid\wr\ resp\ N\ \alpha$$

$$M[\theta 1] \simeq N[\theta 1]$$

$$\beta$$

$subst_{x.\ IdA(M(x),N(x))}\ \alpha\ \beta$

$\equiv (resp\ N\ \alpha)\ \circ\ \beta\ \circ\ (resp\ M\ \alpha)^{-1}$

need inverses, composition, resp

Computational interpretation

1. Define each type by members and paths, with Refl $^{-1}$ and o satisfying groupoid laws

2. Define $\text{subst}_C\ \alpha\ M$ for each C

3. Define resp F α for each F and α

4. Define full Id-elim rule J using subst

Resp

$$\frac{F : A \to B \qquad \alpha : Id_A(M,N)}{resp\ F\ \alpha : Id_B(F\ M, F\ N)}$$

To be a family of terms over A, F must have an associated operation resp F α

Functionality (à la NuPRL) becomes functoriality

Resp for (M,N)

Id(θ1,θ2)

Want resp (**x.(M(x),N(x))**) α:

(M[θ1],N[θ1]) ≃ (M[θ2],N[θ2])

Have

Resp for (M,N)

Want resp (**x.(M(x),N(x))**) α :

Id(θ1,θ2)

(M[θ1],N[θ1]) ≃ (M[θ2],N[θ2])

Have

resp **M** α : M[θ1] ≃ M[θ2]

Resp for (M,N)

Want resp (**x.(M(x),N(x))**) α :

Id(θ1,θ2)

(M[θ1],N[θ1]) ≃ (M[θ2],N[θ2])

Have

resp **M** α : M[θ1] ≃ M[θ2]

resp **N** α : N[θ2] ≃ N[θ2]

Resp for (M,N)

Want resp (**x.(M(x),N(x))**) α :

Id(θ1,θ2)

(M[θ1],N[θ1]) ≃ (M[θ2],N[θ2])

Have

resp **M** α : M[θ1] ≃ M[θ2]

resp **N** α : N[θ2] ≃ N[θ2]

resp (**x.(M(x),N(x))**) α

≡ pair≃ (resp **M** α) (resp **N** α)

Resp for fst

Want resp (**x:(A*B).fst(x)**) α

fst (M[θ1]) ≃ fst(M[θ2])

Have

Resp for fst

Want resp (**x:(A*B).fst(x)**) α

fst (M[θ1]) ≃ fst(M[θ2])

Have $\alpha : Id_{A*B}(M[\theta 1], M[\theta 2])$

Resp for fst

Want resp (**x:(A*B).fst(x)**) α

fst (M[θ1]) ≃ fst(M[θ2])

Have $\alpha : Id_{A*B}(M[\theta 1], M[\theta 2])$

Therefore α ≡ pair≃ α1 α2

Resp for fst

Want resp (**x:(A*B).fst(x)**) α

$$\text{fst}\,(M[\theta 1]) \simeq \text{fst}(M[\theta 2])$$

Have $\alpha : Id_{A*B}(M[\theta 1], M[\theta 2])$

Therefore $\alpha \equiv \text{pair}\simeq\ \alpha 1\ \alpha 2$

$$\text{resp}\,(\mathbf{x.fst(x)})\,(\text{pair}\simeq\ \alpha 1\ \alpha 2)$$
$$\equiv \alpha 1$$

Resp for fst

Want resp (**x:(A*B).fst(x)**) α

fst (M[θ1]) ≃ fst(M[θ2])

Have $\alpha : Id_{A*B}(M[\theta 1], M[\theta 2])$

Therefore α ≡ pair≃ α1 α2

resp (**x.fst(x)**) (pair≃ α1 α2)

≡ α1

2D β redex

Computational interpretation

1. Define each type by members and paths, with Refl $^{-1}$ and o satisfying groupoid laws

2. Define $subst_C$ α M for each C

3. Define resp F α for each F and α

4. **Define full Id-elim rule J using subst**

J from subst

Fix A : type, M : A

$$\frac{C : (\Sigma x{:}A.\mathrm{Id}\ M\ x) \to \mathrm{type} \quad \alpha : \mathrm{Id}_A(M,N) \quad P : C(M,\mathrm{Refl})}{J_B\, P\ \alpha : C(N,\alpha)}$$

Paths from M with free endpoint are inductively generated by (M,Refl)

J from subst

$$\frac{C' : A' \to \text{type} \quad \alpha' : \mathrm{Id}_{A'}(M',N') \quad P' : C'[M']}{\mathrm{subst}_{C'}\ \alpha'\ P' : C'[N']}$$

Fix A : type, M : A

$$\frac{C : (\Sigma x{:}A.\mathrm{Id}\ M\ x) \to \text{type} \quad \alpha : \mathrm{Id}_A(M,N) \quad P : C(M,\mathrm{Refl})}{J_B\ P\ \alpha : C(N,\alpha)}$$

Paths from M with free endpoint are inductively generated by (M,Refl)

J from subst

$$\frac{C' : A' \to \text{type} \quad \alpha' : \mathrm{Id}_{A'}(M',N') \quad P' : C'[M']}{\mathrm{subst}_{C'}\ \alpha'\ P' : C'[N']}$$

Fix A : type, M : A

$$\frac{C : (\Sigma x{:}A.\mathrm{Id}\ M\ x) \to \text{type} \quad \alpha : \mathrm{Id}_{A}(M,N) \quad P : C(M,\mathrm{Refl})}{J_B\ P\ \alpha : C(N,\alpha)}$$

Paths from M with free endpoint are inductively generated by (M,Refl)

- Take A’ = Σx:A.Id M x

J from subst

$$\frac{C' : (\Sigma x{:}A.\mathrm{Id}\ M\ x) \to \mathrm{type} \quad \alpha' : \mathrm{Id}_{\Sigma x:A.\mathrm{Id}\ M\ x}(M',N') \quad P' : C'[M']}{\mathrm{subst}_{C'}\ \alpha'\ P' : C'[N']}$$

Fix A : type, M : A

$$\frac{C : (\Sigma x{:}A.\mathrm{Id}\ M\ x) \to \mathrm{type} \quad \alpha : \mathrm{Id}_A(M,N) \quad P : C(M,\mathrm{Refl})}{J_B\ P\ \alpha : C(N,\alpha)}$$

Paths from M with free endpoint are inductively generated by (M,Refl)

- Take A' = Σx:A.Id M x
- Take C' = C, M' = (M, Refl), N' = (N, α), P' = P

J from subst

$$\frac{C : (\Sigma x{:}A.\mathrm{Id}\ M\ x) \to \mathrm{type} \quad \alpha' : \mathrm{Id}_{\Sigma x:A.\mathrm{Id}\ M\ x}(M,\mathrm{Refl})(N,\alpha) \quad P : C(M,\mathrm{Refl})}{\mathrm{subst}_C\ \alpha'\ P : C(N,\alpha)}$$

Fix A : type, M : A

$$\frac{C : (\Sigma x{:}A.\mathrm{Id}\ M\ x) \to \mathrm{type} \quad \alpha : \mathrm{Id}_A(M,N) \quad P : C(M,\mathrm{Refl})}{J_B\ P\ \alpha : C(N,\alpha)}$$

Paths from M with free endpoint are inductively generated by (M,Refl)

- Take $A' = \Sigma x{:}A.\mathrm{Id}\ M\ x$
- Take $C' = C$, $M' = (M, \mathrm{Refl})$, $N' = (N, \alpha)$, $P' = P$

Therefore suffices to have $\alpha' : \mathrm{Id}_{\Sigma x:A.\mathrm{Id}\ M\ x}(M,\mathrm{Refl})(N,\alpha)$

[Awodey]

J from subst

Given $\alpha : Id_A(M,N)$

want $\alpha' : Id_{\Sigma x:A.Id\ M\ x}(M,Refl)(N,\alpha)$

But have pairing for Id_Σ

$$\frac{\alpha : Id_A(M,M') \qquad \beta : Id_{B[N']}(subst_B\ \alpha\ N,\ N')}{pair{\simeq}\ \alpha\ \beta : Id_{\Sigma x:A.B}(M,N)(M',N')}$$

:B[M] :B[M']

J from subst

Given $\alpha : Id_A(M,N)$

want $\alpha' : Id_{\Sigma x:A.Id\ M\ x}(M,Refl)(N,\alpha)$

$= (pair\simeq\ \alpha\ Refl)$

But have pairing for Id_Σ

$$\frac{\alpha : Id_A(M,M') \qquad \beta : Id_{B[N']}(subst_B\ \alpha\ N,\ N')}{pair\simeq\ \alpha\ \beta : Id_{\Sigma x:A.B}(M,N)(M',N')}$$

:B[M] :B[M']

J from subst

Given $\alpha : Id_A(M,N)$

want $\alpha' : Id_{\Sigma x:A.Id\ M\ x}(M,Refl)(N,\alpha)$

$= (pair\simeq\ \alpha\ Refl)$

But have pairing for Id_Σ

$$\frac{\alpha : Id_A(M,M') \qquad \beta : Id_{B[N']}(subst_B\ \alpha\ N,\ N')}{pair\simeq\ \alpha\ \beta : Id_{\Sigma x:A.B}(M,N)(M',N')}$$

:B[M] :B[M']

and computation rule for subst at Id:

$subst_{x:A.Id\ M\ x}\ \alpha\ Refl$

$\equiv \alpha\ o\ Refl$

$\equiv \alpha$

Technical details

1. Need simultaneous subst for many variables
   to generalize $\mathrm{subst}_{\to}$ to subst_{Π}:
   explicit substitution calculus

2. Judgemental formulation:
   Path judgement $\alpha : M \simeq N$ comes first,
   subst and resp are judgemental concepts.
   $\simeq$ internalized as Id type.

3. "Definitional" equality includes the equations of GPD (e.g. interchange law, functoriality of subst), though simpler axiomatizations are possible using Id type and equality reflection [Garner]

Examples

- Once you know to look for them, subst/resp reductions show up often
- Often apply in situations where it's hard to find a fibration to use J with

Programming:

- Code reuse via isomorphism

Math:

- Algebraic topology: $\pi_1(S^1)$ is Z

Monoid

Monoid : U → U
Monoid X = Σ ⊙ : X → X → X.
Σ u : X.
(Πx,y,z. Id (x ⊙ (y ⊙ z)) ((x ⊙ y) ⊙ z))
* (Πx.Id (x ⊙ u) x)
* (Πx.Id (u ⊙ x) x)

Monoid : type → type
Monoid X = Σ ⊙:X→X→X. Σ u:X. ...

Given (f,f^{-1},α,β) : HEquiv(A, B)
and (⊙,u,…) : Monoid(A)
make (⊙’,u’,…) : Monoid(B)

By hand:

Monoid : type → type
Monoid X = Σ ⊙:X→X→X. Σ u:X. ...

Given (f,f^{-1},α,β) : HEquiv(A, B)
and (⊙,u,…) : Monoid(A)
make (⊙',u',…) : Monoid(B)

By hand:

⊙' = λ y_1,y_2:B . f ((f^{-1} y_1) ⊙ (f^{-1} y_2))
u' = f u

Monoid : type → type
Monoid X = Σ ⊙:X→X→X. Σ u:X. ...

Given (f,f^{-1},α,β) : HEquiv(A, B)
and (⊙,u,…) : Monoid(A)
make (⊙',u',…) : Monoid(B)

By hand:

⊙' = λ y_1,y_2:B . f ((f^{-1} y_1) ⊙ (f^{-1} y_2))
u' = f u

(y ⊙' u') ≃ f ((f^{-1} y) ⊙ (f^{-1} (f u)))
≃ f (f^{-1} y ⊙ u) by α
≃ f (f^{-1} y) by unit law for ⊙ and u
≃ y by β

Automatically!

$$\alpha = ua\,(f, f^{-1}, \alpha, \beta) : A \simeq B$$

Monoid(A)

: Monoid(B)

Automatically!

$\alpha = ua\ (f, f^{-1}, \alpha, \beta) : A \simeq B$

Monoid(A)

$subst_{X:U.Monoid(X)}\ \alpha\ (\odot, u, \ldots)$ **: Monoid(B)**

$\equiv (\lambda y_1, y_2.\ f\ (\ (f^{-1}\ y_1) \odot (f^{-1}\ y_2)\ )$
$\quad f\ u,\ \ldots)$

Automatically!

$\alpha = ua\ (f, f^{-1}, \alpha, \beta) : A \simeq B$

Monoid(A)

$subst_{X:U.Monoid(X)}\ \alpha\ (\odot, u, \ldots)$ **: Monoid(B)**

$\equiv subst_{X:U.\Sigma \odot : X \to X \to X.\Sigma u:X.\ldots}\ \alpha\ (\odot, u, \ldots)$

$\equiv (\lambda y_1, y_2.\ f\ (\ (f^{-1}\ y_1) \odot (f^{-1}\ y_2)\ )$
$\quad f\ u,\ \ldots)$

Automatically!

$\alpha = ua\ (f, f^{-1}, \alpha, \beta) : A \simeq B$

Monoid(A)

$subst_{X:U.Monoid(X)}\ \alpha\ (\odot, u, \ldots)$ **: Monoid(B)**

$\equiv subst_{X:U.\Sigma \odot:X \to X \to X.\Sigma u:X.\ldots}\ \alpha\ (\odot, u, \ldots)$

$\equiv (subst_{x.x \to x \to x}\ \alpha\ \odot,$
$subst_{x.x}\ \alpha\ u, \ldots)$

$\equiv (\lambda y_1, y_2.\ f\ (\ (f^{-1}\ y_1) \odot (f^{-1}\ y_2)\ )$
$f\ u, \ldots)$

Automatically!

$\alpha = ua\ (f, f^{-1}, \alpha, \beta) : A \simeq B$

Monoid(A)

$subst_{X:U.Monoid(X)}\ \alpha\ (\odot, u, \ldots)$ **: Monoid(B)**

$\equiv subst_{X:U.\Sigma \odot:X \to X \to X.\Sigma u:X.\ldots}\ \alpha\ (\odot, u, \ldots)$

$\equiv (subst_{x.x \to x \to x}\ \alpha\ \odot,$
$\quad subst_{x.x}\ \alpha\ u, \ldots)$

$\equiv (\lambda y_1, y_2.\ subst_{x.x}\ \alpha\ (\ (subst_{x.x}\ \alpha^{-1}\ y_1) \odot (subst_{x.x}\ \alpha^{-1}\ y_2)\ )$
$\quad subst_{x.x}\ \alpha\ u, \ldots)$

$\equiv (\lambda y_1, y_2.\ f\ (\ (f^{-1}\ y_1) \odot (f^{-1}\ y_2)\ )$
$\quad f\ u, \ldots)$

Examples

Programming:

- Code reuse via isomorphism

Math:

- Algebraic topology: $\pi_1(S^1)$ is Z (HoTT blog)

Shulman, 2011

simplification Licata, 2012

Work in progress

Conjectured algorithm for 2D:

- working on normalization judgement based on above explanation

Challenges for higher dimensions:

- What does the operational semantics need to be for $\mathsf{Id}_{\mathsf{Id}}\,\mathsf{Id}_{\mathsf{Id}_{\mathsf{Id}}}$ ... ?
- Some computation steps are Id not ≡

where constructive higher-dimensional models will help

Constructivity

Open problem: Can you normalize a program that is written using univalence?

So far:

- Canonicity up to ≡ for 2D case
- Working on normalization algorithm for 2D
- With some minor adjustments, steps valid for low dimensions in full HoTT

Thanks for listening!

Reducibility

Reducibility groupoid $\langle \Gamma \rangle$ for Γ

- Objects: a set of eq. cls. of substitutions $\ll\theta\gg : \Gamma$
- Maps: a set of $\ll\delta\gg : \ll\theta_1\gg \to_{\langle\Gamma\rangle} \ll\theta_2\gg$ s.t. $\delta : \theta_1 \simeq \theta_2 : \Gamma$

Similarly for closed types, A

Reducibility functor $F : \langle A \rangle \to \langle B \rangle$ for $x : A \vdash M : B$

- Objects: $F(\ll N\gg) = \ll[N/x]M\gg$ for all objects $\ll N\gg$ in $\langle A\rangle$
- Maps: $F(\ll\alpha\gg) = \ll \mathsf{resp}_B(\alpha)(M)\gg$ for every $\ll\alpha\gg : \ll N_1\gg \to_{\langle A\rangle} \ll N_2\gg$

Reducibility

Reducibility family $F : \langle \Gamma \rangle \to \mathbf{GPD}$ for $\Gamma \vdash A$

- $F(\ll\theta_1\gg)$ is a red. gpd. for $A[\theta_1]$
- $F(\ll\delta\gg)$ is a red. func. for $x : A[\theta_1] \vdash \mathsf{map}_A(\delta)(x) : A[\theta_2]$, for each $\ll\delta\gg : \ll\theta_1\gg \to_{\langle A \rangle} \ll\theta_2\gg$

Interpretation of Types

Define $[\![\Gamma]\!]$ by induction:

- $[\![\bullet]\!] = \mathbf{1}$;
- $[\![\Gamma, x{:}A]\!] = \int_{[\![\Gamma]\!]} [\![A]\!]$: formal Grothendieck construction

Define family over $[\![\Gamma]\!]$ for $[\![\Gamma \vdash A]\!]$ by induction:

- $[\![\Gamma \vdash \mathsf{set}]\!] = \mathbf{const}(\mathbf{set})$: a universe of sets
- $[\![\Gamma \vdash \mathsf{El}(M)]\!] = \mathbf{elt}[M_*]$: discrete groupoids
- $[\![\Gamma \vdash \Pi x{:}A.B]\!] = \mathbf{pi}_{[\![\Gamma \vdash A]\!]}[\![\Gamma, x{:}A \vdash B]\!]$: dep. exponential

Canonicity

Everything in sight is reducible and preserves reducible equivalences. For example,

If Γ ctx, then $[\![\Gamma]\!]$ is a red. gpd. for Γ.

If $\Gamma \vdash A$ type, then $[\![\Gamma \vdash A]\!]$ is a red. fam. for A over $[\![\Gamma]\!]$.

If $\Gamma \vdash M : A$, then

- $\ll M[\theta_1] \gg \in \mathsf{Ob}\,[\![\Gamma \vdash A]\!](\ll \theta_1 \gg)$ for all $\ll \theta_1 \gg \in \mathsf{Ob}\,[\![\Gamma]\!]$.
- $\ll M[\delta] \gg : \ll \mathsf{map}_A(\delta)(M[\theta_1]) \gg \to_{[\![\Gamma \vdash A]\!](\theta_2)} \ll M[\theta_2] \gg$, whenever $\ll \delta \gg : \ll \theta_1 \gg \to_{[\![\Gamma]\!]} \ll \theta_2 \gg$.