New Extensional Type Theory

Andrew Polonsky

May 12, 2014

λe

$$
\begin{aligned}
A, t, e ::= {} & *_n \mid x \mid \Pi x{:}A.B \mid \Sigma x{:}A.B \mid A \simeq B \mid a \sim_e b \\
& \mid \lambda x{:}A.t \mid st \mid (s, t) \mid \pi_1 t \mid \pi_2 t \\
& \mid *^* \mid \Pi^*[x, x', x^*]{:}A^*.B^* \mid \Sigma^*[x, x', x^*]{:}A^*.B^* \mid \simeq^* A^* B^* \\
& \mid \mathsf{r}(t) \mid \looparrowdownleft_t \mid e(t) \mid \bar{e}(t) \mid t_e \mid t^e
\end{aligned}
$$

λe

$$
\begin{aligned}
A, t, e ::= {} & *_n \mid x \mid \Pi x{:}A.B \mid \Sigma x{:}A.B \\
& \mid \lambda x{:}A.t \mid s t \mid (s, t) \mid \pi_1 t \mid \pi_2 t
\end{aligned}
$$

λe

$$
\begin{aligned}
A, t, e ::= {} & *_n \mid x \mid \Pi x{:}A.B \mid \Sigma x{:}A.B \mid A \simeq B \mid a \sim_e b \\
& \mid \lambda x{:}A.t \mid st \mid (s, t) \mid \pi_1 t \mid \pi_2 t \\
& \mid *^* \mid \Pi^*[x, x', x^*]{:}A^*.B^* \mid \Sigma^*[x, x', x^*]{:}A^*.B^* \mid \simeq^* A^* B^*
\end{aligned}
$$

λe

$$
\begin{aligned}
A, t, e ::= {} & *_n \mid x \mid \Pi x{:}A.B \mid \Sigma x{:}A.B \mid A \simeq B \mid a \sim_e b \\
& \mid \lambda x{:}A.t \mid st \mid (s, t) \mid \pi_1 t \mid \pi_2 t \\
& \mid *^* \mid \Pi^*[x, x', x^*]{:}A^*.B^* \mid \Sigma^*[x, x', x^*]{:}A^*.B^* \mid \simeq^* A^* B^* \\
& \mid \mathsf{r}(t) \mid \looparrowdownleft_t
\end{aligned}
$$

λe

$$
\begin{aligned}
A, t, e ::= {} & *_n \mid x \mid \Pi x{:}A.B \mid \Sigma x{:}A.B \mid A \simeq B \mid a \sim_e b \\
& \mid \lambda x{:}A.t \mid st \mid (s, t) \mid \pi_1 t \mid \pi_2 t \\
& \mid *^* \mid \Pi^*[x, x', x^*]{:}A^*.B^* \mid \Sigma^*[x, x', x^*]{:}A^*.B^* \mid \simeq^* A^* B^* \\
& \mid \mathrm{r}(t) \mid \looparrowdownleft_t \mid e(t) \mid \bar{e}(t) \mid t_e \mid t^e
\end{aligned}
$$

λe

$$
\begin{aligned}
A, t, e ::= {} & *_n \mid x \mid \Pi x{:}A.B \mid \Sigma x{:}A.B \mid A \simeq B \mid a \sim_e b \\
& \mid \lambda x{:}A.t \mid st \mid (s, t) \mid \pi_1 t \mid \pi_2 t \\
& \mid *^* \mid \Pi^*[x, x', x^*]{:}A^*.B^* \mid \Sigma^*[x, x', x^*]{:}A^*.B^* \mid \simeq^* A^* B^* \\
& \mid \mathsf{r}(t) \mid \circlearrowleft_t \mid e(t) \mid \bar{e}(t) \mid t_e \mid t^e
\end{aligned}
$$

Outline

- Extensionality: the problem.
- Extensionality and dependent types.
- The system $\lambda\simeq$.
- The system λe.

Outline

- Extensionality: the problem.
- Extensionality and dependent types.
- The system $\lambda\simeq$.
- The system λe.

Extensionality

- A central difficulty of formalizing constructive mathematics in type theory is that the equality relation is intensional: two objects are only considered equal if they can be converted into one another by a finite sequence of local syntactic transformations.
- A given function could be implemented by two different algorithms; even if they give the same input–output behavior, they would be considered different objects in type theory.

Equality in type theory

- The Martin-Löf identity type Id_A reifies the conversion relation into the type structure. It is intensional, and the ground type $\mathit{Id}_{\mathbb{N}\to\mathbb{N}}(\lambda n.n+1)(\lambda n.1+n)$ is not inhabited.

Equality in type theory

- The Martin-Löf identity type Id_A reifies the conversion relation into the type structure. It is intensional, and the ground type $Id_{\mathbb{N}\to\mathbb{N}}(\lambda n.n{+}1)(\lambda n.1{+}n)$ is not inhabited.
- Martin-Löf proposed to reflect this type back into the conversion relation, so that type-theoretic constructions could be used in the proofs that two terms are convertible. This choice leads to type theory becoming undecidable.

Equality in type theory

- The Martin-Löf identity type Id_A reifies the conversion relation into the type structure. It is intensional, and the ground type $Id_{\mathbb{N}\to\mathbb{N}}(\lambda n.n+1)(\lambda n.1+n)$ is not inhabited.
- Martin-Löf proposed to reflect this type back into the conversion relation, so that type-theoretic constructions could be used in the proofs that two terms are convertible. This choice leads to type theory becoming undecidable.
- Voevodsky proposed to add Univalence Axiom which is a form of universe extensionality and implies function extensionality. Without computational interpretation, assuming this axiom leads to the failure of canonicity property.

What is extensional equality?

- Things are extensionally equal if they appear the same "on the outside". A more precise statement of this intuition is: extensional equality is concerned with how things are observed, how they can be used.
- In particular, the extensional equality associated to a given type constructor should be given in terms of elimination forms for that type.

$$
\begin{aligned}
f \simeq_{A \to B} g &= \Pi x : A.\ fx \simeq_B gx \\
f \simeq_{A \to B} g &= \Pi x x' : A.\ x \simeq_A x' \to fx \simeq_B gx' \\
(a, b) \simeq_{A \times B} (a', b') &= (a \simeq_A a') \times (b \simeq_B b') \\
p \simeq_{A \times B} p' &= (\pi_1 p \simeq_A \pi_1 p') \times (\pi_2 p \simeq_B \pi_2 p')
\end{aligned}
$$

$$
\begin{aligned}
f \simeq_{A \to B} g \quad &= \quad \Pi x x' : A.\ x \simeq_A x' \to f x \simeq_B g x' \\
p \simeq_{A \times B} p' \quad &= \quad (\pi_1 p \simeq_A \pi_1 p') \times (\pi_2 p \simeq_B \pi_2 p')
\end{aligned}
$$

- If two terms of type T are extensionally equal after applying every possible eliminator to the type, then the two terms are extensionally equal at that type.
- Equality should also form a (higher-dimensional) equivalence relation, and be preserved by every construction of type theory (substitution of equals-for-equals).

Coquand's axioms

$$
\begin{array}{rrcl}
(a : A) & \mathrm{r}(a) & : & a \simeq_A a \\
(x : A \vdash B(x) : *) & \texttt{transp} & : & B(a) \to (a \simeq_A a') \to B(a') \\
(b : B(a)) & \texttt{Jcomp} & : & \texttt{transp}\, b\, \mathrm{r}(a) \simeq_{B(a)} b \\
(a : A) & \pi_a & : & \texttt{isContr}(\Sigma x{:}A.a \simeq_A x) \\
\text{FA:} & \multicolumn{3}{l}{(\Pi x{:}A)\ fx \simeq_{B(x)} gx \ \to\ f \simeq_{\Pi x:A.B(x)} g}
\end{array}
$$

Voevodsky has shown that the last axiom is implied by

UA: The canonical map $A \simeq_* B \to \mathrm{Weq} AB$ is an equivalence

Our plan

1. Define $a \simeq_A a'$ by induction on A making sure it is a congruence with respect to all constructions of type theory:
$$\frac{x : A \vdash t(x) : T \qquad \vdash a^* : a \simeq_A a'}{\vdash t(a^*) : t(a) \simeq_T t(a')}$$
2. By taking $x \notin \mathrm{FV}(t)$, get $t() : t \simeq_T t$ to define $\mathrm{r}(t)$.
3. Get `transp` from
$$a \simeq_A a' \to B(a) \simeq_* B(a')$$
by adding operators for transporting back and forth along $B(a^*) : B(a) \simeq_* B(a')$.
4. Use these same operators for higher-dimensional analogues of symmetry and transitivity (the Kan filling conditions).

$$\frac{x : A \vdash t(x) : T \qquad \vdash a^* : a \simeq_A a'}{\vdash t(a^*) : t(a) \simeq_T t(a')}$$

- In dependent type theory, the types of $b(a)$ and $b(a')$ might be different:

$$\begin{array}{l} \Gamma, x : A \vdash B(x) : * \\ \Gamma, x : A \vdash b(x) : B(x) \end{array} \qquad \frac{\Gamma \vdash a : A}{\Gamma \vdash b(a) : B(a)}$$

- If $(a, b), (a', b') \in \Sigma x{:}A.B(x)$, then we can have $a^* : a \simeq_A a'$, but b and b' cannot be compared directly: $b : B(a)$, $b' : B(a')$, and $B(a) \neq B(a')$.
- To reason about extensional equality in the dependent setting, we need a notion of dependent equality.

Outline

- Extensionality: the problem.
- Extensionality and dependent types.
- The system $\lambda\simeq$.
- The system λe.

Dependent equality

- When $a^* : a \simeq_A a'$, and $x : A \vdash b(x) : B(x)$, we first consider $B(a^*) : B(a) \simeq B(a')$.
- Assumption.
  Every type equality $e : T \simeq T'$ induces a relation

$$\sim e : T \to T' \to *$$

- In particular, for $B(a^*) : B(a) \simeq B(a')$, we have

$$\sim B(a^*) : B(a) \to B(a') \to *$$

- We now type $b(a^*)$ as $\sim B(a^*) b(a) b(a')$, which we write as

$$b(a^*) : b(a) \sim_{B(a^*)} b(a')$$

The relation on the universe

- We want to define an equality relation on every type by induction on type structure, and we want to prove that every term of type theory preserves this relation.
- The definition of the system will set out by assuming that there is a binary relation on the universe of all types, and that every type constructor preserves this relation
(the relation is a congruence wrt type structure).
- This relation is denoted by $A \simeq B$. It is a new type constructor.
- The eliminator of this type is the relation $\sim e : A \to B \to *$.
- The constructors are the congruence axioms.

$\lambda{\simeq}$

$$A, t, e ::= *_n \mid x \mid \Pi x{:}A.B \mid \Sigma x{:}A.B \mid A \simeq B \mid a \sim_e b$$
$$\mid \lambda x{:}A.t \mid st \mid (s, t) \mid \pi_1 t \mid \pi_2 t$$
$$\mid *^* \mid \Pi^*[x, x', x^*]{:}A^*.B^* \mid \Sigma^*[x, x', x^*]{:}A^*.B^* \mid \simeq^* A^* B^*$$

$$\frac{A : *_n \qquad B : *_n}{A \simeq B : *_n} \qquad \frac{e : A \simeq B \qquad a : A \qquad b : B}{a \sim_e b : *_n}$$

- We have $*_n^* : *_n \simeq *_n$.
- If $A^* : A \simeq A'$, and $x^* : x \sim_{A^*} x' \vdash B^* : B \simeq B'$, then $\Pi x{:}A.B \simeq \Pi x'{:}A'.B'$, and $\Sigma x{:}A.B \simeq \Sigma x'{:}A'.B'$.
- If $A^* : A \simeq A'$ and $B^* : B \simeq B'$, then $\simeq^* A^* B^* : (A \simeq B) \simeq (A' \simeq B')$.

$\lambda\simeq$

$$\begin{aligned} A, t, e ::= {} & *_n \mid x \mid \Pi x{:}A.B \mid \Sigma x{:}A.B \mid A \simeq B \mid a \sim_e b \\ & \mid \lambda x{:}A.t \mid st \mid (s,t) \mid \pi_1 t \mid \pi_2 t \\ & \mid *^* \mid \Pi^*[x,x',x^*]{:}A^*.B^* \mid \Sigma^*[x,x',x^*]{:}A^*.B^* \mid \simeq^* A^* B^* \end{aligned}$$

The logical conditions are captured by the rewrite rules:

$$\begin{aligned} f \sim_{\Pi^*[x,x',x^*]:A^*.B^*} f' \quad & \longrightarrow \quad \prod_{a:A} \prod_{a':A'} \prod_{a^*:a \sim_{A^*} a'} fa \sim_{B^*(a,a',a^*)} f'a' \\ (a,b) \sim_{\Sigma^*[x,x',x^*]:A^*.B^*} (a',b') \quad & \longrightarrow \quad \sum_{a^*:a \sim_{A^*} a'} b \sim_{B^*(a,a',a^*)} b' \\ e \sim_{\simeq^* A^* B^*} e' \quad & \longrightarrow \quad \prod \begin{pmatrix} a : A \\ a' : A' \\ a^* : a \sim_{A^*} a' \end{pmatrix} \prod \begin{pmatrix} b : B \\ b' : B' \\ b^* : b \sim_{B^*} b' \end{pmatrix} \\ & \qquad\qquad (a \sim_e b) \simeq (a' \sim_{e'} b') \\ A \sim_{*^*} B \quad & \longrightarrow \quad A \simeq B \end{aligned}$$

Theorem

Suppose $\Gamma \vdash t(x_1, \ldots, x_n) : T(x_1, \ldots, x_n)$, where

$$\Gamma = x_1 : A_1, \ldots, x_n : A_n(x_1, \ldots, x_{n-1})$$

There exists a term $t^* = t(x_1^*, \ldots, x_n^*)$ such that

$$\begin{pmatrix} x_1 : A_1 & \cdots & x_n : A_n(x_1, \ldots, x_{n-1}) \\ x_1' : A_1 & \cdots & x_n' : A_n(x_1', \ldots, x_{n-1}') \\ x_1^* : x_1 \sim_{A_1^*} x_1' & \cdots & x_n^* : x_n \sim_{A_n^*} x_n' \end{pmatrix}$$

$$\vdash t^* : t(x_1, \ldots, x_n) \sim_{T^*} t(x_1', \ldots, x_n')$$

In particular, for a closed term $\vdash t : T$, there are closed terms

$$\begin{array}{rcccl} \mathrm{r}(t) & := & t^* & : & t \sim_{\mathrm{r}(T)} t \\ \mathrm{r}(T) & := & T^* & : & T \sim_{\mathrm{r}(*)} T \\ \mathrm{r}(*_n) & := & *^* & : & *_n \sim_{\mathrm{r}(*_{n+1})} *_n \end{array}$$

The extensional identity type

- For a closed type A, the type equality $\mathsf{r}(A) : A \simeq A$ is the identity equivalence on A.
- The relation $A^{\simeq} : A \to A \to *$ associated to this equivalence is the extensional identity type on A. It is denoted as

$$a \simeq_A a' \quad := \quad a \sim_{\mathsf{r}(A)} a'$$

Outline

- Extensionality: the problem.
- Extensionality and dependent types.
- The system $\lambda\simeq$.
- The system λe.

λe

$$
\begin{aligned}
A, t, e ::= {} & *_n \mid x \mid \Pi x{:}A.B \mid \Sigma x{:}A.B \mid A \simeq B \mid a \sim_e b \\
& \mid \lambda x{:}A.t \mid st \mid (s,t) \mid \pi_1 t \mid \pi_2 t \\
& \mid *^* \mid \Pi^*[x,x',x^*]{:}A^*.B^* \mid \Sigma^*[x,x',x^*]{:}A^*.B^* \mid \simeq^* A^* B^* \\
& \mid \mathsf{r}(t) \mid \circlearrowleft_t \mid e(t) \mid \bar{e}(t) \mid t_e \mid t^e
\end{aligned}
$$

$$
\frac{a : A}{\mathsf{r}(a) : a \sim_{\mathsf{r}(A)} a}
$$

$$
\frac{e : A \simeq B \qquad a : A}{\begin{array}{rcl} e(a) & : & B \\ a_e & : & a \sim_e e(a) \end{array}}
\qquad
\frac{e : A \simeq B \qquad b : B}{\begin{array}{rcl} \bar{e}(b) & : & A \\ b^e & : & \bar{e}(b) \sim_e b \end{array}}
$$

Higher substitution

The system admits higher-dimensional cell substitution operations.
In the one-dimensional case, it is typed as follows:

$$\frac{\begin{array}{rl} \Gamma, x_1 : A_1, \ldots, x_n : A_n & \vdash t : T \\ \Gamma & \vdash a_1^* : a_1 \sim_{\mathsf{r}(A_1)} a_1' \\ \Gamma & \vdash a_2^* : a_2 \sim_{A_2[a_1^*//x_1]} a_2' \\ & \vdots \\ \Gamma & \vdash a_n^* : a_n \sim_{A_n[a_1^*,\ldots,a_{n-1}^*//x_1,\ldots,x_{n-1}]} a_n' \end{array}}{\Gamma \vdash t[a_1^*, \ldots, a_n^*//x_1, \ldots, x_n] : t[\vec{a}/\vec{x}] \sim_{T[\vec{a}^*//\vec{x}]} t[\vec{a}'/\vec{x}]}$$

Example

We can define the `mapOnPaths` operator

$$\frac{\Gamma \vdash f : \Pi x{:}A.B \qquad \Gamma \vdash a^* : a \simeq_A a'}{\Gamma \vdash f.a^* : fa \sim_{B[a^*//x]} fa'}$$

It is defined by taking

$$f.a^* \quad := \quad \mathsf{r}(f)aa'a^*$$

which computes as

$$(\lambda x{:}A.t).a^* \quad = \quad t[a^*//x]$$

Composition and Symmetry

Let $\alpha : a_1 \simeq_A a_2$.

$$\mathring{\alpha}(x) \quad : \quad (x \simeq_A a_1) \simeq (x \simeq_A a_2) \qquad \alpha_\circ(y) \quad : \quad (a_1 \simeq_A y) \simeq (a_2 \simeq_A y)$$
$$\mathring{\alpha}(x) \quad := \quad (x \simeq_A y)[\alpha//y] \qquad \alpha_\circ(y) \quad := \quad (x \simeq_A y)[\alpha//x]$$

Let $a_{01} : a_0 \simeq_A a_1$. Let $a_{23} : a_2 \simeq_A a_3$.

$$\alpha^\circ a_{01} \quad : \quad a_0 \simeq_A a_2 \qquad \alpha_\circ a_{23} \quad : \quad a_1 \simeq_A a_3$$
$$\alpha^\circ a_{01} \quad := \quad \mathring{\alpha}(a_0)(a_{01}) \qquad \alpha_\circ a_{23} \quad := \quad \overline{\alpha_\circ(a_3)}(a_{23})$$

(Also, $a_{01\circ}\alpha : a_0 \simeq_A a_2$ and $a_{23}{}^\circ\alpha : a_1 \simeq_A a_3$.)

$$\overline{\alpha} := \overline{\mathring{\alpha}(a_2)}(\mathsf{r}(a_2)) \quad : \quad a_2 \simeq_A a_1$$
$$\underline{\alpha} := \alpha_\circ(a_1)(\mathsf{r}(a_1)) \quad : \quad a_2 \simeq_A a_1$$

Proving the axioms

- $\mathrm{r}(a) : a \sim_{\mathrm{r}(A)} a :\equiv a \simeq_A a$

Proving the axioms

- $\mathsf{r}(a) : a \sim_{\mathsf{r}(A)} a := a \simeq_A a$
- $\mathtt{transp}_{B(x)}\ b_0\ a_{01} := B(a_{01})(p)$, where

$$B(a_{01}) = B[a_{01}//x] : B(a_0) \simeq B(a_1)$$

Proving the axioms

- $\mathsf{r}(a) : a \sim_{\mathsf{r}(A)} a := a \simeq_A a$
- $\mathtt{transp}_{B(x)}\ b_0\ a_{01} := B(a_{01})(p)$, where

$$B(a_{01}) = B[a_{01}//x] : B(a_0) \simeq B(a_1)$$

- $\mathtt{transp}_{B(x)}\ b\ \mathsf{r}(a) := B(\mathsf{r}(a))(b) = \mathsf{r}(B(a))(b)$, since

$$t[\mathsf{r}(a)//x] = \mathsf{r}(t(a))$$

  always holds. Now $b_{\mathsf{r}(B(a))} : b \simeq_{B(a)} \mathsf{r}(B(a))(b)$.

Proving the axioms

- $\mathsf{r}(a) : a \sim_{\mathsf{r}(A)} a :\equiv a \simeq_A a$
- $\mathtt{transp}_{B(x)}\; b_0\; a_{01} :\equiv B(a_{01})(p)$, where

$$B(a_{01}) = B[a_{01}/\!/x] : B(a_0) \simeq B(a_1)$$

- $\mathtt{transp}_{B(x)}\; b\; \mathsf{r}(a) :\equiv B(\mathsf{r}(a))(b) = \mathsf{r}(B(a))(b)$, since

$$t[\mathsf{r}(a)/\!/x] = \mathsf{r}(t(a))$$

  always holds. Now $b_{\mathsf{r}(B(a))} : b \simeq_{B(a)} \mathsf{r}(B(a))(b)$.
- Given $a : A$, for any $\alpha : a \simeq_A x$, put

$$p_{x,\alpha} :\equiv (\mathsf{r}(\mathsf{r}(A))\, a\, a\, \mathsf{r}(a)\, a\, x\, \alpha)(\mathsf{r}(a))$$
$$P_{x,\alpha} :\equiv \mathrel{\rlap{\downarrow}{\multimap}}{\mathsf{r}(a)}_{\mathsf{r}(\mathsf{r}(A))\, a\, a\, \mathsf{r}(a)\, a\, x\, \alpha}$$

  Then $\lambda x \lambda \alpha.(p_{x,\alpha}, P_{x,\alpha})$ shows $(a, \mathsf{r}(a))$ to be a center of contraction of type $\Sigma x{:}A.a \simeq_A x$.

Proving the axioms

- $\mathsf{r}(a) : a \sim_{\mathsf{r}(A)} a := a \simeq_A a$
- $\texttt{transp}_{B(x)}\ b_0\ a_{01} := B(a_{01})(p)$, where

$$B(a_{01}) = B[a_{01}//x] : B(a_0) \simeq B(a_1)$$

- $\texttt{transp}_{B(x)}\ b\ \mathsf{r}(a) := B(\mathsf{r}(a))(b) = \mathsf{r}(B(a))(b)$, since

$$t[\mathsf{r}(a)//x] = \mathsf{r}(t(a))$$

  always holds. Now $b_{\mathsf{r}(B(a))} : b \simeq_{B(a)} \mathsf{r}(B(a))(b)$.
- Given $a : A$, for any $\alpha : a \simeq_A x$, put

$$p_{x,\alpha} := (\mathsf{r}(\mathsf{r}(A))\, a\, a\, \mathsf{r}(a)\, a\, x\, \alpha)(\mathsf{r}(a))$$
$$P_{x,\alpha} := \looparrowleft_{\mathsf{r}(a)_{\mathsf{r}(\mathsf{r}(A))\, a\, a\, \mathsf{r}(a)\, a\, x\, \alpha}}$$

  Then $\lambda x \lambda \alpha.(p_{x,\alpha}, P_{x,\alpha})$ shows $(a, \mathsf{r}(a))$ to be a center of contraction of type $\Sigma x{:}A.a \simeq_A x$.
- Function extensionality: by construction.

Conclusion

- We have defined a type system with a natural type-theoretic construction of the extensional equality type.
- We conjecture that the system satisfies strong normalization and hence has decidable type checking.
- The system provides a lambda calculus for computing with higher cells.
- Future work includes univalence, higher inductive types, and homotopy reflection principles.