A Strongly Normalizing Computation Rule for Univalence in Higher-Order Propositional Logic

Robin Adams Marc Bezem Thierry Coquand

TYPES 2016, Novi Sad, Serbia
May 26 2016

This talk is a literate Agda file:
`https://www.github.com/radams78/Univalence`

Introduction

Type Theory Orthodoxy

- To enjoy a good meaning explanation, a type theory should enjoy these properties:
  - **Confluence** — Reduction is confluent.
  - **Strong Normalization** — Every reduction strategy terminates.
  - **Canonicity** — Hence every well-typed term of type A reduces to a unique canonical form of A.
    - E.g. every closed term of type $\mathbb{N}$ reduces to a unique numeral.
- The *univalence axiom* postulates a function

$$\mathit{isotoid} : A \simeq B \to A = B$$

  that is an inverse to the obvious function $A = B \to A \simeq B$.
- This breaks canonicity.

Possible Solutions

- Lower our standards
  - **Voevodsky's Conjecture — Propositional Canonicity**
    For every closed term $t : \mathbb{N}$, there exists a numeral n and a proof $\vdash p : Id_{\mathbb{N}}(t, n)$.
- Use a type theory in which *isotoid* is definable (Cubical Type Theory, [Pol14])
- Introduce a reduction rule for *isotoid*.

Our Approach

We begin with a small type theory, and work our way up to the full HoTT.

1. λoe — **Predicative Higher-Order Propositional Logic**
   A type theory with:
   - a universe Ω of *propositions* with $\bot$ and $\supset$
   - a universe U of *small types* with Ω and $\rightarrow$
   - for any two terms $M, N : A$, a (large) type $M =_A N$.
2. λoi — **P.H.O.P.L. with Equality**
   Make $\delta =_\phi \epsilon$ a proposition. (So we can form propositions like $M =_A N \supset N =_A M$.)

For the future: universal quantification, natural numbers, inductive types, path elimination, . . .

About the Formalization

About the Formalization

This work is being formalized in Agda (work in progress).
It will involve several systems and reduction relations.
I want to prove only once:

- $M[x := N][y := P] \equiv M[y := P[x := N]][x := N]$
- If $M \twoheadrightarrow N$ then $M[x := P] \twoheadrightarrow N[x := P]$
- If $M[x := P]$ is SN then M is SN.
- Etc.

The formalization includes a general notion of ‘grammar’ and ‘reduction relation’. (To do: general notion of derivation rules.)

Grammars

Example: Simply-typed lambda calculus

$$\begin{array}{lll} \text{Type} & A & ::= & \bot \mid A \to A \\ \text{Term} & M & ::= & x \mid \lambda x : A.M \mid MM \end{array}$$

- Two kinds: 'Type' (non-variable kind) and 'Term' (variable kind)
- Four constructors:
  - $\bot$ — kind Type
  - $\to$ — kind (Type, Type) Type
  - λ — kind (Type, (Term) Term) Term
  - *app* — kind (Term, Term) Term

Grammars

A *grammar* over a taxonomy consists of: consists of:

- a set of *expression kinds*;
- a subset of expression kinds, called the *variable kinds*. We refer to the other expession kinds as *non-variable kinds*.
- a set of *constructors*, each with an associated *constructor kind* of the form

$$((A_{11},\ldots,A_{1r_1})B_1,\ldots,(A_{m1},\ldots,A_{mr_m})B_m)C \qquad (1)$$

  where each A_{ij} is a variable kind, and each B_i and C is an expression kind.
- a function assigning, to each variable kind K, an expression kind, the *parent* of K.

A *taxonomy* consists of:

- a set of *expression kinds*;
- a subset of expression kinds, called the *variable kinds*. We refer to the other expession kinds as *non-variable kinds*.

```
record Taxonomy : Set₁ where
  field
    VarKind : Set
    NonVarKind : Set

  data ExpressionKind : Set where
    varKind : VarKind → ExpressionKind
    nonVarKind : NonVarKind → ExpressionKind
```

We can now define the set of expressions over a grammar:

```
data Subexpression : Alphabet → ∀ C → Kind C → Set
Expression : Alphabet → ExpressionKind → Set
Body : Alphabet → ∀ {K} → Kind (-Constructor K) → Set

Expression V K = Subexpression V -Expression (base K)
Body V {K} C = Subexpression V (-Constructor K) C

infixr 50 _,,_
data Subexpression where
  var : ∀ {V} {K} → Var V K → Expression V (varKind K)
  app : ∀ {V} {K} {C} → Constructor C → Body V {K} C →
    Expression V K

  out : ∀ {V} {K} → Body V (out K)
  _,,_ : ∀ {V} {K} {A} {L} {C} → Expression (extend V A) L →
    Body V {K} C → Body V (Π A L C)
```

Predicative Higher-Order Propositional Logic

The Simply-Typed Lambda Calculus

We begin with the simply-typed lambda calculus (no surprises so far):

$$\begin{array}{lrcl} \text{Type} & A & ::= & \Omega \mid A \rightarrow A \\ \text{Term} & M, \phi & ::= & x \mid \lambda x : A.M \mid MM \end{array}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : A \rightarrow B} \qquad \frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

Propositional Logic

Ω is the universe of propositions:

$$\begin{array}{lrcl} \text{Term} & M, \phi & ::= & \cdots \mid \bot \mid \phi \supset \phi \\ \text{Proof} & \delta & ::= & p \mid \lambda p : \phi . \delta \mid \delta\delta \end{array}$$

$$\frac{\Gamma \vdash \delta : \phi \to \psi \quad \Gamma \vdash \epsilon : \phi}{\Gamma \vdash \delta\epsilon : \psi} \qquad \frac{\Gamma, p : \phi \vdash \delta : \psi}{\Gamma \vdash \lambda p : \phi . \delta : \phi \to \psi}$$

$$\frac{\Gamma \vdash \delta : \phi \quad \Gamma \vdash \psi : \Omega}{\Gamma \vdash \delta : \psi} \ (\phi \simeq \psi)$$

Extensional Equality

On top of this we add extensional equality.

$$\begin{array}{llll} \text{Path} & P & ::= & e \mid \text{ref}\,(M) \mid \text{univ}_{\phi,\phi}\,(P,P) \mid P \supset^* P \mid \\ & & & P_{NN}P \mid \lambda\!\!\lambda\!\!\lambda e : x =_A x.P \\ \text{Proof} & \delta & ::= & \cdots \mid P^+ \mid P^- \end{array}$$

Judgement form $\Gamma \vdash P : M =_A N$.
Two main ways to prove equality:

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \text{ref}\,(M) : M =_A M} \qquad \frac{\Gamma \vdash \delta : \phi \to \psi \quad \Gamma \vdash \epsilon : \psi \to \phi}{\Gamma \vdash \text{univ}_{\phi,\psi}\,(\delta,\epsilon) : \phi =_\Omega \psi}$$

We can eliminate equalities in Ω:

$$\frac{\Gamma \vdash P : \phi =_\Omega \psi}{\Gamma \vdash P^+ : \phi \to \psi} \qquad \frac{\Gamma \vdash P : \psi =_\Omega \psi}{\Gamma \vdash P^- : \psi \to \phi}$$

Congruence rule for λ:

$$\frac{\Gamma, x : A, y : A, e : x =_A y \vdash Mx =_B Ny}{\Gamma \vdash \lambda\!\!\lambda\!\!\lambda e : x =_A y . P : M =_{A \to B} N}$$

e, x and y are bound within P.
Congruence rules and conversion

$$\frac{\Gamma \vdash P : \phi =_\Omega \phi' \quad \Gamma \vdash Q : \psi =_\Omega \psi'}{\Gamma \vdash P \supset^* Q : \phi \supset \psi =_\Omega \phi' \supset \psi'}$$

$$\frac{\Gamma \vdash P : M =_{A \to B} M' \quad \Gamma \vdash Q : N =_A N'}{\Gamma \vdash P_{NN'} Q : MN =_B M'N'}$$

$$\frac{\Gamma \vdash P : M =_A N \quad \Gamma \vdash M' : A \quad \Gamma \vdash N' : A}{\Gamma \vdash P : M' =_A N'} \; (M \simeq M', N \simeq N')$$

The Reduction Relation

The 'β-rules':

$$(\lambda x : A.M)N \rhd M[x := N] \qquad (\lambda p : \phi.\delta)\epsilon \rhd \delta[p := \epsilon]$$

$$\mathsf{ref}\,(\phi)^+ \rhd \lambda p : \phi.p \qquad \mathsf{ref}\,(\phi)^- \rhd \lambda p : \phi.p$$

$$\mathsf{univ}_{\phi,\psi}\,(\delta,\epsilon)^+ \rhd \delta \qquad \mathsf{univ}_{\phi,\psi}\,(\delta,\epsilon)^- \rhd \epsilon$$

The Reduction Relation

We make univ and ref move out past $\supset^*$ and application:

$$\mathsf{ref}\,(\phi) \supset^* \mathsf{univ}_{\psi,\chi}\,(\delta,\epsilon) \rhd \mathsf{univ}_{\phi\supset\psi,\phi\supset\chi}(\lambda p, q.\delta(pq), \lambda p, q.\epsilon(pq))$$

$$\mathsf{univ}_{\phi,\psi}\,(\delta,\epsilon) \supset^* \mathsf{ref}\,(\chi) \rhd \mathsf{univ}_{\phi\supset\chi,\psi\supset\chi}\,(\lambda p, q.p(\epsilon q), \lambda p, q.p(\delta q))$$

$$\mathsf{univ}_{\phi,\psi}\,(\delta,\epsilon) \supset^* \mathsf{univ}_{\phi',\psi'}\,(\delta',\epsilon')$$
$$\rhd \mathsf{univ}_{\phi\supset\phi',\psi\supset\psi'}\,(\lambda p, q.\delta'(p(\epsilon q)), \lambda p, q.\epsilon'(p(\delta q)))$$

$$\mathsf{ref}\,(\phi) \supset^* \mathsf{ref}\,(\psi) \rhd \mathsf{ref}\,(\phi \supset \psi) \qquad \mathsf{ref}\,(M)_{N_1 N_2}\,\mathsf{ref}\,(N) \rhd \mathsf{ref}\,(MN)$$

The Reduction Relation

We construct a proof of $M =_{A \to B} N$, then apply it. What is the result?

- $\mathsf{ref}\,(M)_{N_1 N_2}\,\mathsf{ref}\,(N) \rhd \mathsf{ref}\,(MN)$
- $(\lambda\!\!\lambda\!\!\lambda e : x =_A y.P)_{N_1 N_2} Q \rhd P[x := N_1, y := N_2, e := Q]$
- If $P \not\equiv \mathsf{ref}\,(-)$, then $\mathsf{ref}\,(\lambda x : A.M)_{NN'}\, P \rhd ???$

$$\Gamma, x : A \vdash M : B, \qquad \Gamma \vdash P : N =_A N'$$

Path Substitution

Define the operation of *path substitution* such that, if $P : M =_A M'$ then $N\{x := P : M \sim M'\} \equiv N\{x := P\} : N[x := M] =_B N[x := M']$.

$$
\begin{aligned}
x\{x := P\} &\stackrel{\text{def}}{=} P \\
y\{x := P\} &\stackrel{\text{def}}{=} \text{ref}\,(y) \qquad (y \not\equiv x) \\
\bot\{x := P\} &\stackrel{\text{def}}{=} \text{ref}\,(\bot) \\
(LL')\{x := P : M \sim M'\} & \\
&\stackrel{\text{def}}{=} L\{x := P\}_{L'[x:=M]L'[x:=M']} L'\{x := P\} \\
(\lambda y : A.L)\{x := P\} & \\
&\stackrel{\text{def}}{=} \lambda\!\!\lambda\!\!\lambda e : a =_A a'.L\{x := P, y := e : a \sim a'\} \\
(\phi \supset \psi)\{x := P\} &\stackrel{\text{def}}{=} \phi\{x := P\} \supset^* \psi\{x := P\}
\end{aligned}
$$

The Reduction Relation

We construct a proof of $M =_{A \to B} N$, then apply it. What is the result?

- $\text{ref}\,(M)\,\text{ref}\,(N) \rhd \text{ref}\,(MN)$
- $(\lambda\!\!\lambda\!\!\lambda e : x =_A y.P)_{MN} Q \rhd P[x := M, y := N, e := Q]$
- If $P \not\equiv \text{ref}\,(-)$, then $\text{ref}\,(\lambda x : A.M)_{N,N'}\, P \rhd M\{x := P : N \sim N'\}$

Confluence

Theorem (Local Confluence)

The reduction relation $\longrightarrow$ is locally confluent. That is, if $E \longrightarrow F$ and $E \longrightarrow G$, then there exists H such that $F \twoheadrightarrow H$ and $G \twoheadrightarrow H$.

Proof.

Case analysis on $E \longrightarrow F$ and $E \longrightarrow G$. There are no critical pairs. □

$$\begin{aligned}&\text{Local-Confluent} : \forall\ \{V\}\ \{C\}\ \{K\}\\ &\quad \{E\ F\ G : \text{Subexpression}\ \ V\ C\ K\} \to E \Rightarrow F \to E \Rightarrow G \to\\ &\quad \Sigma[\ H \in \text{Subexpression}\ \ V\ C\ K\ ]\ (F \twoheadrightarrow H \times G \twoheadrightarrow H)\end{aligned}$$

Corollary (Newman's Lemma)

Every strongly normalizing term is confluent, hence has a unique normal form.

Strong Normalization

Tait's Method

We define a model of the type theory with types as sets of terms.
For every type (proposition, equation) A in context Γ, define the set of *computable* terms $E_\Gamma(A)$.
The definition is such that:

1. If $M \in E_\Gamma(A)$ then $\Gamma \vdash M : A$ and M is strongly normalizing.
2. $E_\Gamma(A)$ is closed under *key redex expansion*.
3. If $A \simeq B$ then $E_\Gamma(A) = E_\Gamma(B)$.

Define the sets of *computable* terms, proofs and paths as follows.

$$
\begin{aligned}
E_\Gamma(\Omega) &\stackrel{\text{def}}{=} \{M \mid \Gamma \vdash M : \Omega, M \in \mathbf{SN}\} \\
E_\Gamma(A \to B) &\stackrel{\text{def}}{=} \{M \mid \Gamma \vdash M : A \to B, \\
&\qquad \forall(\Delta \supseteq \Gamma)(N \in E_\Delta(A)).MN \in E_\Delta(B), \\
&\qquad \forall(\Delta \supseteq \Gamma)(N, N' \in E_\Delta(A))(P \in E_\Delta(N =_A N')). \\
&\qquad\quad \mathsf{ref}\,(M)_{NN'}\, P \in E_\Gamma(MN =_B MN')\}
\end{aligned}
$$

Computable Terms

$$E_\Gamma(\bot) \stackrel{\text{def}}{=} \{\delta \mid \Gamma \vdash \delta : \bot, \delta \in \mathbf{SN}\}$$

$$E_\Gamma(\phi \to \psi) \stackrel{\text{def}}{=} \{\delta \mid \Gamma \vdash \delta : \phi \to \psi,$$

$$\forall(\Delta \supseteq \Gamma)(\epsilon \in E_\Delta(\phi)).\delta\epsilon \in E_\Gamma(\psi)\}$$

$$E_\Gamma(\phi) \stackrel{\text{def}}{=} \{\delta \mid \Gamma \vdash \delta : \bot, \delta \in \mathbf{SN}\}$$

(ϕ neutral)

$$E_\Gamma(\phi) \stackrel{\text{def}}{=} E_\Gamma(nf(\phi))$$

(ϕ a normalizable term of type Ω)

Computable Terms

$$E_\Gamma(\phi =_\Omega \psi) \stackrel{\text{def}}{=} \{P \mid \Gamma \vdash P : \phi =_\Omega \psi, \\ P^+ \in E_\Gamma(\phi \to \psi), P^- \in E_\Gamma(\psi \to \phi)\}$$

$$E_\Gamma(M =_{A \to B} M') \stackrel{\text{def}}{=} \{P \mid \Gamma \vdash P : M =_{A \to B} M', \\ \forall(\Delta \supseteq \Gamma)(N, N' \in E_\Delta(A))(Q \in E_\Delta(N =_A N')). \\ P_{NN'} Q \in E_\Delta(MN =_B M'N')\}$$

The Main Theorem

Theorem

Let σ be a substitution such that, for all $x : A \in \Gamma$, we have $\sigma(x) \in E_\Delta(A)$. Then, if $\Gamma \vdash M : A$, then $M[\sigma] \in E_\Delta(A)$.

```
Computable-Sub : ∀ {U} {V} {K} (σ : Sub U V) {Γ} {Δ}
    {M : Expression U (varKind K)} {A} →
    σ : Γ ⇒C Δ → Γ ⊢ M : A → valid Δ → E' Δ (A [ σ ]) (M [ σ ])
```

Corollary (Strong Normalization)

Every well-typed term, proof and path is strongly normalizing.

```
Strong-Normalization : ∀ V K (Γ : Context V)
    (M : Expression V (varKind K)) A → Γ ⊢ M : A → SN M
```

Corollary (Consistency)

There is no proof δ such that $\vdash \delta : \bot$.

The System λoi

Internal Equality

We place the propositions $M =_A N$ inside Ω, so we can form (and prove!)

$$\text{sym} : M =_A N \supset N =_A M, \quad \text{trans} : M =_A N \supset N =_A P \supset M =_A P$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : A}{\Gamma \vdash M =_A N : \Omega}$$

$$\frac{\Gamma \vdash \delta : M =_A M' \quad \Gamma \vdash \epsilon : N =_A N'}{\Gamma \vdash \delta =^*_A \epsilon : (M =_A N) =_\Omega (M' =_A N')}$$

New reductions include:

$$\begin{aligned} & \text{ref}\,(\phi) =^*_\Omega \text{univ}_{\psi,\chi}\,(\delta, \epsilon) \\ \triangleright & \text{univ}_{\phi =_\Omega \psi, \phi =_\Omega \chi}\left(\lambda p : \phi =_\psi .\text{univ}_{\phi,\chi}\left(\lambda q : \phi.\delta(p^+ q), \lambda q : \chi.p^-(\epsilon q)\right),\right. \\ & \left.\lambda p : \phi =_\Omega \chi.\text{univ}_{\phi,\psi}\left(\lambda q : \phi.\epsilon(p^+ q), \lambda q : \psi.p^-(\delta q)\right)\right) \end{aligned}$$

Conclusion

Conclusion

- We have shown two systems that each have all these properties:
  - Univalence
  - Strong Normalization
  - Confluence of typed terms
  - Canonicity
- So it is possible!
- The simplicity is due to the separation between terms and proofs.
- For the future: extract a normalizer. Universal quantification.
- Follow the progress here: `www.github.com/radams78/Univalence`

Reference

Andrew Polonsky.
Internalization of extensional equality.
CoRR, abs/1401.1148, 2014.