Calculus of constructions explained

I.Zhirkov

2015

Outline

What is CC?

- A higher-order formalism for constructive proofs in natural deduction style;

What is CC?

- A higher-order formalism for constructive proofs in natural deduction style;
- Informally: typed λ-calculus, where types are normal terms too;
- Or: high level functional programming language with a type system rich enough to express any specification.

Outline

Basic CC syntax

- Variables (are typed with other terms).
- Application:
  $M\ N$
- λ-abstraction:
  $(\lambda x : T)M$
- Product operator
  $[x : T]M$
  think *forall* or *function type*.
- Special constant *.

About *

- “Type of all types”;
- Type of propositions;
- Products over * are special (are called *contexts*);
- Has no type itself.

Outline

Contexts

We will call some of terms **contexts**.
Contexts are lists.

- Nil = *
- Cons = Product operator

$$* \in C^n$$

$$[x : M]N \in C^n, M \in \Lambda^n, N \in C^{n+1}$$

x may occur in N, not in M
Contexts have no type (cause * has none).
Contexts are type of proposition schemas (think nat->bool->Prop), declaring their parameters.

Terms can be

- Contexts (products over *);
- Objects;

Terms can be

- Contexts (products over *);
- Objects;
  - Objects of type * are **usual types** aka **propositions**;
  - Other objects.

Terms can be

- Contexts (products over *);
- Objects;
  - Objects of type * are **usual types** aka **propositions**;
  - Other objects.

$$\Lambda = C \cup O$$

$$\Lambda^n = C^n \cup O^n$$

$$O = \bigcup_{n \geq 0} O^n, \quad C = \bigcup_{n \geq 0} C^n$$

[variables' de Bruijn indices] $\leq n$
Λ denotes all terms;
C or Λ_o – contexts;
O or Λ_l – objects.

Objects

- Variables (think de Bruijn indexes)

$$k \in O^n, \text{ if } 1 \leq k \leq n$$

- Product

$$[x : M]N \in O^n, \text{ if } M \in \Lambda^n, N \in O^{n+1}$$

- Abstraction

$$(\lambda x : M)N \in O^n, \text{ if } M \in \Lambda^n, N \in O^{n+1}$$

- Application

$$(M\ N) \in O^n, \text{ if } M, N \in O^n$$

Two relations, two kinds of judgements

- $\Gamma \vdash \Delta$
  Δ *is valid in a valid context* Γ
- $\Gamma \vdash M : N$
  In valid Γ, M is well-typed of type N

Inference

Contexts:

$$\frac{}{\vdash *}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma[x : \Delta] \vdash *}$$

$$\frac{\Gamma \vdash M : *}{\Gamma[x : M] \vdash *}$$

Inference

Contexts:

$$\frac{}{\vdash *}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma[x : \Delta] \vdash *}$$

$$\frac{\Gamma \vdash M : *}{\Gamma[x : M] \vdash *}$$

Product:

$$\frac{\Gamma[x : M] \vdash \Delta}{\Gamma \vdash [x : M]\Delta}$$

$$\frac{\Gamma[x : M_1] \vdash M_2 : *}{\Gamma \vdash [x : M_1]M_2 : *}$$

Inference

- Variables ($I \leq |\Gamma|$)

$$\frac{\Gamma \vdash *}{\Gamma \vdash I : \Gamma/I}$$

- Abstraction

$$\frac{\Gamma[x : M_1] \vdash M_2 : P}{\Gamma \vdash (\lambda x : M_1)M_2 : [x : M_1]P}$$

- Application

$$\frac{\Gamma \vdash M : [x : P]Q \quad \Gamma \vdash N : P}{\Gamma \vdash (MN) : [N/x]Q}$$

this is substitution $N \mapsto 1$*,being var idx*

Note on predicativity

*A **predicative** system enforces the constraint that, when an object is defined using some sort of quantifier, none of the quantifiers may ever be instantiated with the object itself.*
Avoid naive set theory paradoxes.

Note on predicativity

*A **predicative** system enforces the constraint that, when an object is defined using some sort of quantifier, none of the quantifiers may ever be instantiated with the object itself.*
Avoid naive set theory paradoxes.

- Martin-Loef type theory has a hierarchy of type universes (predicative):

$$([A : U_i]A) : U_{i+1}$$

Note on predicativity

*A **predicative** system enforces the constraint that, when an object is defined using some sort of quantifier, none of the quantifiers may ever be instantiated with the object itself.*
Avoid naive set theory paradoxes.

- Martin-Loef type theory has a hierarchy of type universes (predicative):

$$([A : U_i]A) : U_{i+1}$$

- Calculus of constructions has only one universe $*$ and is nonpredicative:

$$([A : *]A) : *$$

  $*$ has no type however.

Note on predicativity

- Adding $\overline{* \vdash * : *}$ results in Girard paradox (each type is inhabited).

Outline

Third kind of judgements

- $\Gamma \vdash \Delta$
  Δ is valid in a valid context Γ
- $\Gamma \vdash M : N$
  In valid Γ, M is well-typed of type N
- $\Gamma \vdash M \cong N$
  M is convertible to N in Γ.
  Smallest congruence over terms w.r.t. β reduction.

Kinds of calculus of constructions

- Restricted – β-reduction only on 'logical' level (in types)
- Full – β-reduction on both type level and term level is allowed.

Restricted's only use is a simplier proof of decidability of three judgements.

β-reduction rules for restricted CoC

Symmetry, transitivity of $\cong$;
Δ *is a context*

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta \cong \Delta}$$

β-reduction rules for restricted CoC

Symmetry, transitivity of $\cong$;
Δ *is a context*

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta \cong \Delta}$$

$$\frac{\Gamma \vdash M : \Delta}{\Gamma \vdash M \cong M}$$

β-reduction rules for restricted CoC

Symmetry, transitivity of $\cong$;
Δ *is a context*

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta \cong \Delta}$$

$$\frac{\Gamma \vdash M : \Delta}{\Gamma \vdash M \cong M}$$

$$\frac{\Gamma \vdash P_1 \cong P_2 \qquad \Gamma[x : P_1] \vdash M_1 \cong M_2}{\Gamma \vdash [x : P_1]\, M_1 \cong [x : P_2] M_2}$$

β-reduction rules for restricted CoC

Symmetry, transitivity of $\cong$;
Δ *is a context*

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta \cong \Delta}$$

$$\frac{\Gamma \vdash M : \Delta}{\Gamma \vdash M \cong M}$$

$$\frac{\Gamma \vdash P_1 \cong P_2 \qquad \Gamma[x : P_1] \vdash M_1 \cong M_2}{\Gamma \vdash [x : P_1]\, M_1 \cong [x : P_2] M_2}$$

$$\frac{\Gamma \vdash P_1 \cong P_2 \qquad \Gamma[x : P_1] \vdash M_1 \cong M_2 \qquad \Gamma[x : P_1] \vdash M_1 : \Delta_1}{\Gamma \vdash (\lambda x : P_1) M_1 \cong (\lambda x : P_2) M_2}$$

β-reduction rules for restricted CoC - 2

$$\frac{\Gamma \vdash (M\ N) : \Delta \qquad \Gamma \vdash M \cong M_1}{\Gamma \vdash (M\ N) \cong (M_1\ N)}$$

$$\frac{\Gamma \vdash (M\ N) : \Delta \qquad \Gamma \vdash N \cong N_1}{\Gamma \vdash (M\ N) \cong (M\ N_1)}$$

$$\frac{\Gamma[x : P] \vdash M : \Delta \qquad \Gamma \vdash N : P}{\Gamma \vdash ((\lambda x : P)M\ N) \cong [N/x]M}$$

β-reduction only for “logical” redexes.

$$\frac{\Gamma \vdash M : P \qquad \Gamma \vdash P \cong Q}{\Gamma \vdash M : Q}$$

For restricted CoC decidability of judgements is proven independently from normalization theorem.

β-reduction rules for full CoC-1

Symmetry, transitivity of $\cong$;

$$\frac{\Gamma \vdash M : N}{\Gamma \vdash M \cong M}$$

$$\frac{\Gamma \vdash P_1 \cong P_2 \qquad \Gamma[x : P_1] \vdash M_1 \cong M_2}{\Gamma \vdash [x : P_1]\, M_1 \cong [x : P_2] M_2}$$

β-reduction rules for full CoC-1

Symmetry, transitivity of $\cong$;

$$\frac{\Gamma \vdash M : N}{\Gamma \vdash M \cong M}$$

$$\frac{\Gamma \vdash P_1 \cong P_2 \qquad \Gamma[x : P_1] \vdash M_1 \cong M_2}{\Gamma \vdash [x : P_1]\ M_1 \cong [x : P_2] M_2}$$

$$\frac{\Gamma \vdash P_1 \cong P_2 \qquad \Gamma[x : P_1] \vdash M_1 \cong M_2 \qquad \Gamma[x : P_1] \vdash M_1 : N_1}{\Gamma \vdash (\lambda x : P_1) M_1 \cong (\lambda x : P_2) M_2}$$

β-reduction rules for full CoC-2

$$\frac{\Gamma \vdash (M\ N) : P \qquad \Gamma \vdash M \cong M_1 \qquad \Gamma \vdash N \cong N_1}{\Gamma \vdash (M\ N) \cong (M_1\ N_1)}$$

$$\frac{\Gamma[x : A] \vdash M : P \qquad \Gamma \vdash N : A}{\Gamma \vdash ((\lambda x : A)M\ N) \cong [N/x]M}$$

Comparison-1

- Restricted

$$\frac{\Gamma \vdash M : \Delta}{\Gamma \vdash M \cong M}$$

- Full

$$\frac{\Gamma \vdash M : N}{\Gamma \vdash M \cong M}$$

Comparison-2

- Restricted

$$\frac{\Gamma \vdash P_1 \cong P_2 \qquad \Gamma[x : P_1] \vdash M_1 \cong M_2 \qquad \Gamma[x : P_1] \vdash M_1 : \Delta_1}{\Gamma \vdash (\lambda x : P_1)M_1 \cong (\lambda x : P_2)M_2}$$

- Full

$$\frac{\Gamma \vdash P_1 \cong P_2 \qquad \Gamma[x : P_1] \vdash M_1 \cong M_2 \qquad \Gamma[x : P_1] \vdash M_1 : N_1}{\Gamma \vdash (\lambda x : P_1)M_1 \cong (\lambda x : P_2)M_2}$$

Comparison-3

- Restricted

$$\frac{\Gamma \vdash (M\ N) : \Delta \qquad \Gamma \vdash M \cong M_1}{\Gamma \vdash (M\ N) \cong (M_1\ N)}$$

$$\frac{\Gamma \vdash (M\ N) : \Delta \qquad \Gamma \vdash N \cong N_1}{\Gamma \vdash (M\ N) \cong (M\ N_1)}$$

- Full

$$\frac{\Gamma \vdash (M\ N) : P \qquad \Gamma \vdash M \cong M_1 \qquad \Gamma \vdash N \cong N_1}{\Gamma \vdash (M\ N) \cong (M_1\ N_1)}$$

Comparison-4

- Restricted

$$\frac{\Gamma[x : P] \vdash M : \Delta \qquad \Gamma \vdash N : P}{\Gamma \vdash ((\lambda x : P)M\ N) \cong [N/x]M}$$

- Full

$$\frac{\Gamma[x : A] \vdash M : P \qquad \Gamma \vdash N : A}{\Gamma \vdash ((\lambda x : A)M\ N) \cong [N/x]M}$$

Outline

Stripping

- We can get rid of specification and extract untyped λ-term, representing computation.
- Contexts get rid of quantifications over contexts;
  Why? A context is a type of proposition schemas, its inhabitants are not part of computation process.

Stripping for $\Gamma \vdash M : Obj$

Context arity α_Γ – how many elements of list are objects;
Canonical injection $j_\Gamma : \{1 \ldots \alpha_\Gamma\} \to \Gamma$ – indices of objects in Γ.
Stripped algorithm $\nu_\Gamma(M)$:

- Variable:
  $\nu_\Gamma(k) := j_\Gamma^{-1}(k)$
- Application:
  $\nu_\Gamma((M : \_)(N : \mathrm{Ctx})) := \nu_\Gamma(M)$
  $\nu_\Gamma((M : \_)(N : \mathrm{Obj})) := \nu_\Gamma(M)\ \nu_\Gamma(N)$
- Abstraction:
  $$\begin{cases} \nu_\Gamma((\lambda x : \mathrm{Obj}_1\ N : \mathrm{Obj}_2) := \lambda x.\nu_\Delta(N), & \Delta = \Gamma[x : \mathrm{Obj}_1] \\ \nu_\Gamma((\lambda x : \mathrm{Ctx}\ \ N : \mathrm{Obj}) := \nu_\Delta(N), & \Delta = \Gamma[x : \mathrm{Ctx}] \end{cases}$$

Outline

Interpretation of constructions

- Take all closed terms, add special constant:

$$\mathscr{I} = \Lambda^0 \cup \{\Omega\}$$

- $A \subset \mathscr{I}$ is **saturated** iff:

  1. $\Omega \in A$
  2. $SN(b_1) \dots SN(b_n) \Rightarrow (\Omega b_1 \dots b_n) \in A$
  3. all elements in A are SN
  4. $SN(b) \Rightarrow (([b/x]a)b_1 \dots b_n) \in A \Rightarrow (((\lambda x.a)b)b_1 \dots b_n) \in A$

- $\mathscr{U}$ – all saturated subsets of $\mathscr{I}$

SN denotes *strongly normalisable*

Dependent product

Definition

If $A \in \mathscr{U}$, $F \in \mathscr{I} \to \mathscr{U}$, then
$\Pi(A, F) := \{t \in \mathscr{I} | \forall x \in A : \ (t\ x) \in F(x)\}$

$\mathscr{I}$ are programs, $\mathscr{U}$ are types, $\mathscr{I} \to \mathscr{U}$ are dependent types.

Lemma

$\mathscr{U}$ is closed under arbitrary intersection and union.

Ω is only needed during proof.

Functionality

Definition (restricted calculus)

$\phi(\text{Object}) := \mathscr{I}$
$\phi(*) := \mathscr{U}$
$\phi([x : P]\Gamma) = \phi(P) \to \phi(\Gamma)$ (set of all such functions)

Definition (full calculus)

$\phi(\text{Object}) := \mathscr{I}$
$\phi(*) := \mathscr{U}$
$\phi([x : \Delta]\Gamma) = \phi(\Delta) \to \phi(\Gamma)$
$\phi([x : P]\Gamma) = \{f \in \phi(P) \to \phi(\Gamma) | t \cong u \Rightarrow f(t) = f(u)\}$
Why? cause arbitrary β-reduction is allowed.

Environment

Definition

For $\Gamma = [x_n : A_n] \cdots [x_1 : A_1]*$ **environment** is a product:
$\Phi(\Gamma) = \phi(A_n) \times \cdots \times \phi(A_1)$

$\Gamma \vdash M : \mathrm{Obj}$

Interpretation: $\rho_\Gamma(M) : \Phi(\Gamma) \to \phi(\mathrm{Obj})$
$\nu_\Gamma(M)$ is an α_Γ-ary function $\mathscr{I}^{\alpha_\Gamma} \to \mathscr{I}$ (maps arguments to its α_Γ free variables).
recall that α_Γ is context arity.
$\pi_\Gamma : \Phi(\Gamma) \to \mathscr{I}^{\alpha_\Gamma}$ is defined by type forgetting j_Γ.
Interpretation: $\rho_\Gamma(M) = \nu_\Gamma(M) \circ \pi_\Gamma$
“untype M and forget about quantifications over contexts”

$\Gamma \vdash M : \mathrm{Ctx}$

$\rho_\Gamma(M) : \Phi(\Gamma) \to \phi(\mathrm{Ctx})$

- Product:
  - $\Gamma \vdash [x : \mathrm{Ctx}]M : *$
    $f : \Phi(\Gamma) \times \phi(\mathrm{Ctx}) \to \mathscr{U}$
    $f := \rho_{\Gamma[x:\mathrm{Ctx}]}(M)$

$$\rho_\Gamma([x : \mathrm{Ctx}]M) := a \mapsto \bigcap \{f(a, x) | x \in \phi(\mathrm{Ctx})\}, \quad a \in \Phi(\Gamma)$$

  - $\Gamma \vdash [x : \mathrm{Obj}]M : *$
    $f : \Phi(\Gamma) \to \mathscr{U}, \qquad g : \Phi(\Gamma) \times \mathscr{I} \to \mathscr{U}$
    $f := \rho_{\Gamma[x:\mathrm{Obj}]}(\mathrm{Obj}), \quad g := \rho_{\Gamma[x:\mathrm{Obj}]}(M)$

$$\rho_\Gamma([x : \mathrm{Obj}]M) :- a \mapsto \Pi(f(a), g(a)), \quad a \in \Phi(\Gamma)$$

$\Gamma \vdash M : \mathrm{Ctx}$

$\rho_\Gamma(M) : \Phi(\Gamma) \to \phi(\mathrm{Ctx})$

- Variable: $\rho_\Gamma(x)$ selects x-th variable from context.
- Abstraction:
  $\Gamma \vdash (\lambda x : M)\ N : [x : M]P$
  $f : \Phi(\Gamma) \times \phi(M) \to \phi(P)$
  $f := \rho_{\Gamma[x:M]}(N)$

$$\rho_\Gamma((\lambda x : M)N) := a \mapsto \Big(x \mapsto f(a, x)\Big)$$

- Application
  $\Gamma \vdash (M\ N) : [N/x]P$

$$\rho_\Gamma((M\ N)) := y \mapsto \rho_\Gamma(M)(y, \rho_\Gamma(N, y))$$

Interpretation of contexts

Inductively defined on formation of $\Gamma \vdash *$ (routine).
$D(\Gamma) \hookrightarrow \Phi(\Gamma)$

- $\vdash *$ implies $D(\Gamma) = \Phi(\Gamma) = 1$
- $\Gamma[x : \mathrm{Obj}] \vdash *$ implies
  $D(\Gamma[x : \mathrm{Obj}]) \coloneqq \{(a, x) | a \in D(\Gamma) \wedge x \in \rho_\Gamma(M)(a)\}$
- $\Gamma[x : \mathrm{Ctx}] \vdash *$ implies
  $D(\Gamma[x : \mathrm{Ctx}]) \coloneqq \{(a, x) | a \in D(\Gamma) \wedge x \in \phi(\mathrm{Ctx})\}$
  $= D(\Gamma) \times \phi(\mathrm{Ctx})$

Example

$[A : *][x : A] \vdash *$ is interpreted as $\{(A, x) \in \mathscr{U} \times \mathscr{I} | x \in A\}$.

If $\Gamma \vdash M : N$ and $\Gamma \vdash N : *$, then for all $x \in D(\Gamma)$ pure λ-term $\rho_\Gamma(M, x)$ is an element of saturated set $\rho_\Gamma(P, x)$

Example

If $[A : *][x : A] \vdash x : A$ then with $\Gamma = [A : *][x : A]$ we have $D(\Gamma) = \{(A, x) \in \mathscr{U} \times \mathscr{I} | x \in A\}$.
$[A : *][x : A]\Gamma \vdash x : A$ is interpreted as $f : \mathscr{U} \times \mathscr{I} \to \mathscr{I}$, which maps $(A, x) \mapsto x$.

Outline

-
- $\Gamma \vdash$ *Judgement* is decidable (restricted);
- $\Gamma \vdash M : N$ deducing N from Γ and M is decidable(restricted, full);
- Strongly normalizable (restricted,full).
- $\Gamma \vdash M \cong N \Rightarrow \phi(M) = \phi(N)$

CoC modifications

- Calculus of Constructions;
- Calculus of Constructions with Inductive Definitions;
- Coq v7: Calculus of (Co)Inductive constructions (Cic);
- Coq v8: Predicative Calculus of (Co)Inductive constructions (pCic)

Outline

Utility tweaks

- Global environement;
- Typing rules consider environement;
- Additional reductions for *let... in...* constructs and global definitions;

Outline

Universes and impredicativity

- Coq has many type universes (sorts): *Set*, *Prop*, Type_i for $i \in \mathbb{N}$
- *Set* : Type_0, *Prop* : Type_0, Type_0 : Type_1 etc.
- Since Coq v8 *Set* is predicative by default (unless launched with `-impredicative-set`), so such definitions

  ```
  Definition nat : Set := forall (C:Set), C → (C→ C) → C.
  ```

  are not allowed.

Outline

Inductive definitions

In nonpredicative CoC inductive types are expressed as non first-class objects.

```
Definition nat : Set := forall (C:Set), C → (C → C) → C.
Definition zero : nat := fun C z f ⇒ z.
Definition succ : nat → nat := fun n C z f ⇒ f (n C z f).
```