Semantic consistency proofs for systems of illative combinatory logic

Łukasz Czajka

June 1, 2015

Illative combinatory logic

What is it?

- The original Combinatory *Logic*, but fixed.

Illative combinatory logic

What is it?

- The original Combinatory *Logic*, but fixed.
- Based on a long (but not so popular) line of work initiated by Moses Schönfinkel and Haskell Curry.

Illative combinatory logic

What is it?

- The original Combinatory *Logic*, but fixed.
- Based on a long (but not so popular) line of work initiated by Moses Schönfinkel and Haskell Curry. (continued by Fitch, Hindley, Seldin, Bunder, Dekkers, Barendregt,. . .)

Illative combinatory logic

What is it?

- The original Combinatory *Logic*, but fixed.
- Based on a long (but not so popular) line of work initiated by Moses Schönfinkel and Haskell Curry. (continued by Fitch, Hindley, Seldin, Bunder, Dekkers, Barendregt,. . .)
- A certain *approach* to logic.

Combinatory logic

The general idea

Reduce systems of logic to a certain “simple” form.

Combinatory logic

The general idea

Reduce systems of logic to a certain “simple” form.

- Syntax: only a single binary application operation plus some constants.

Combinatory logic

The general idea

Reduce systems of logic to a certain "simple" form.

- Syntax: only a single binary application operation plus some constants.
- Inference rules: "simple", not involving "complex" notions like substitution.

Naive combinatory logic

A simple (inconsistent) system

- Terms: applicative terms over constants $\supset, \mathsf{K}, \mathsf{S}, \mathsf{Q}$.

Naive combinatory logic

A simple (inconsistent) system

- Terms: applicative terms over constants $\supset, \mathsf{K}, \mathsf{S}, \mathsf{Q}$.
- Judgements: $\Gamma \vdash X$.

Naive combinatory logic

A simple (inconsistent) system

- Terms: applicative terms over constants $\supset, \mathsf{K}, \mathsf{S}, \mathsf{Q}$.
- Judgements: $\Gamma \vdash X$.
- Notational conventions:
  - $X \supset Y \equiv \supset XY$,
  - $X = Y \equiv \mathsf{Q}XY$.

Naive combinatory logic

A simple (inconsistent) system – rules

$$\frac{}{\Gamma, X \vdash X}$$

$$\frac{\Gamma, X \vdash Y}{\Gamma \vdash X \supset Y}\ (\supset_i) \qquad \frac{\Gamma \vdash X \quad \Gamma \vdash X \supset Y}{\Gamma \vdash Y}\ (\supset_e)$$

$$\frac{}{\Gamma \vdash \mathsf{S}XYZ = XZ(YZ)} \qquad \frac{}{\Gamma \vdash \mathsf{K}XY = X}$$

$$\frac{\Gamma \vdash X \quad \Gamma \vdash X = Y}{\Gamma \vdash Y}\ (\mathrm{eq})$$

$+$ usual rules for equality.

Naive combinatory logic

Abstraction $\lambda x.X$ with the property

$$\vdash (\lambda x.X)Y = X[Y/x]$$

is definable using combinators.

Naive combinatory logic

Abstraction $\lambda x.X$ with the property

$$\vdash (\lambda x.X)Y = X[Y/x]$$

is definable using combinators.

It follows that for every term X with x free there exists a term M such that $\vdash M = X[M/x]$.

Naive combinatory logic

Curry's paradox

For an arbitrary given term Y, there is X such that
$\vdash X = (X \supset Y)$.

Naive combinatory logic

Curry's paradox

For an arbitrary given term Y, there is X such that
$\vdash X = (X \supset Y)$.

Using $\vdash X = (X \supset Y)$ one shows $\vdash Y$.

Illative combinatory logic

A simple system

- Terms: untyped λ-terms over constants $\forall, \supset, \mathrm{Prop}, \mathrm{Type}$.

Illative combinatory logic

A simple system

- Terms: untyped λ-terms over constants $\forall, \supset, \mathrm{Prop}, \mathrm{Type}$.
- Notational conventions:
  - $X \supset Y \equiv \supset XY$,
  - $\forall x : X \,.\, Y \equiv \forall X(\lambda x \,.\, Y)$,
  - $X : Y \equiv YX$.

Illative combinatory logic

A simple system – rules

$$\frac{\Gamma, X \vdash Y \quad \Gamma \vdash X : \mathrm{Prop}}{\Gamma \vdash X \supset Y} \qquad \frac{\Gamma \vdash X \quad \Gamma \vdash X \supset Y}{\Gamma \vdash Y}$$

$$\frac{\Gamma, Xx \vdash Yx \quad \Gamma \vdash X : \mathrm{Type} \quad x \notin \mathrm{FV}(\Gamma, X, Y)}{\Gamma \vdash \forall XY} \qquad \frac{\Gamma \vdash \forall XY \quad \Gamma \vdash XZ}{\Gamma \vdash YZ}$$

Illative combinatory logic

A simple system – rules

$$\frac{\Gamma, X \vdash Y \quad \Gamma \vdash X : \mathrm{Prop}}{\Gamma \vdash X \supset Y} \qquad \frac{\Gamma \vdash X \quad \Gamma \vdash X \supset Y}{\Gamma \vdash Y}$$

$$\frac{\Gamma, Xx \vdash Yx \quad \Gamma \vdash X : \mathrm{Type} \quad {\scriptstyle x \notin \mathrm{FV}(\Gamma, X, Y)}}{\Gamma \vdash \forall XY} \qquad \frac{\Gamma \vdash \forall XY \quad \Gamma \vdash XZ}{\Gamma \vdash YZ}$$

$$\frac{\Gamma \vdash X : \mathrm{Prop} \quad \Gamma, X \vdash Y : \mathrm{Prop}}{\Gamma \vdash (X \supset Y) : \mathrm{Prop}}$$

$$\frac{\Gamma, Xx \vdash (Yx) : \mathrm{Prop} \quad \Gamma \vdash X : \mathrm{Type} \quad {\scriptstyle x \notin \mathrm{FV}(\Gamma, X, Y)}}{\Gamma \vdash (\forall XY) : \mathrm{Prop}}$$

Illative combinatory logic

A simple system – rules

$$\frac{\Gamma, X \vdash Y \quad \Gamma \vdash X : \mathrm{Prop}}{\Gamma \vdash X \supset Y} \qquad \frac{\Gamma \vdash X \quad \Gamma \vdash X \supset Y}{\Gamma \vdash Y}$$

$$\frac{\Gamma, Xx \vdash Yx \quad \Gamma \vdash X : \mathrm{Type} \quad {\scriptstyle x \notin \mathrm{FV}(\Gamma, X, Y)}}{\Gamma \vdash \forall XY} \qquad \frac{\Gamma \vdash \forall XY \quad \Gamma \vdash XZ}{\Gamma \vdash YZ}$$

$$\frac{\Gamma \vdash X : \mathrm{Prop} \quad \Gamma, X \vdash Y : \mathrm{Prop}}{\Gamma \vdash (X \supset Y) : \mathrm{Prop}}$$

$$\frac{\Gamma, Xx \vdash (Yx) : \mathrm{Prop} \quad \Gamma \vdash X : \mathrm{Type} \quad {\scriptstyle x \notin \mathrm{FV}(\Gamma, X, Y)}}{\Gamma \vdash (\forall XY) : \mathrm{Prop}}$$

$$\frac{\Gamma \vdash X \quad X =_{\beta\eta} Y}{\Gamma \vdash Y} \qquad \frac{\Gamma \vdash X}{\Gamma \vdash X : \mathrm{Prop}}$$

Induction

For natural numbers

Induction principle **applicable to untyped terms**.

$$\frac{\Gamma \vdash X0 \quad \Gamma, x : \mathrm{Nat}, Xx \vdash X(sx) \quad x \notin FV(\Gamma, X)}{\Gamma \vdash \forall x : \mathrm{Nat} \, . \, Xx}$$

Induction

For natural numbers

Induction principle **applicable to untyped terms**.

$$\frac{\Gamma \vdash X0 \quad \Gamma, x : \mathrm{Nat}, Xx \vdash X(sx) \quad x \notin FV(\Gamma, X)}{\Gamma \vdash \forall x : \mathrm{Nat} \,.\, Xx}$$

The following would be less useful:

$$\forall f : \mathrm{Nat} \to \mathrm{Prop} \,.\, ((f0 \wedge (\forall x : \mathrm{Nat} \,.\, fx \supset f(sx))) \supset \forall x : \mathrm{Nat} \,.\, fx)$$

“Types”

Notational conventions

- $X \to Y \equiv \lambda f.\forall x : X.(fx : Y)$,

“Types”

Notational conventions

- $X \to Y \equiv \lambda f.\forall x : X.(fx : Y)$,
- $(x : X) \to Y(x) \equiv \lambda f.\forall x : X.(fx : Y(x))$,

"Types"

Notational conventions

- $X \to Y \equiv \lambda f.\forall x : X.(fx : Y)$,
- $(x : X) \to Y(x) \equiv \lambda f.\forall x : X.(fx : Y(x))$,
- $\Sigma(x : X)Y(x) \equiv \lambda p.(\pi_1 p : X) \wedge (\pi_2 p : Y(\pi_1 p))$,

"Types"

Notational conventions

- $X \to Y \equiv \lambda f.\forall x : X.(fx : Y)$,
- $(x : X) \to Y(x) \equiv \lambda f.\forall x : X.(fx : Y(x))$,
- $\Sigma(x : X)Y(x) \equiv \lambda p.(\pi_1 p : X) \wedge (\pi_2 p : Y(\pi_1 p))$,
- $\ldots$

“Types”

Postulated formation rules

$$\frac{\Gamma \vdash X : \mathrm{Type} \quad \Gamma, x : X \vdash Y : \mathrm{Type} \quad x \notin \mathrm{FV}(\Gamma, X, Y)}{\Gamma \vdash (X \to Y) : \mathrm{Type}}$$

$$\frac{\Gamma \vdash X : \mathrm{Type} \quad \Gamma, x : X \vdash Y(x) : \mathrm{Type} \quad x \notin \mathrm{FV}(\Gamma, X, Y)}{\Gamma \vdash ((x : X) \to Y(x)) : \mathrm{Type}}$$

$$\frac{\Gamma \vdash X : \mathrm{Type} \quad \Gamma, x : X \vdash Y(x) : \mathrm{Type} \quad x \notin \mathrm{FV}(\Gamma, X, Y)}{\Gamma \vdash (\Sigma(x : X) Y(x)) : \mathrm{Type}}$$

$$\ldots$$

“Types”

Derived rules

$$\frac{\Gamma, x : X \vdash Z : Y \quad \Gamma \vdash X : \mathrm{Type} \quad x \notin \mathrm{FV}(\Gamma, X, Y)}{\Gamma \vdash (\lambda x. Z) : X \to Y}$$

$$\frac{\Gamma \vdash F : X \to Y \quad \Gamma \vdash Z : X}{\Gamma \vdash FZ : Y}$$

"Types"

Derived rules

$$\frac{\Gamma, x : X \vdash Z : Y(x) \quad \Gamma \vdash X : \mathrm{Type} \quad {\scriptstyle x \notin \mathrm{FV}(\Gamma, X, Y)}}{\Gamma \vdash (\lambda x.Z) : (x : X) \to Y(x)}$$

$$\frac{\Gamma \vdash F : (x : X) \to Y(x) \quad \Gamma \vdash Z : X}{\Gamma \vdash FZ : YZ}$$

Using the induction principle(s) one may show that simply-typable (dependently-typable, . . . ?) terms extented with well-founded recursion are still “typable” in our system.

“Types”

Reflection

$$\vdash \forall x, y : \mathrm{TermCode}\,.\, \texttt{checktype}(x, y) \supset (\texttt{eval}(x) : \texttt{eval}(y))$$

Slogan

Type checking/inference algorithms as proof tactics.

Some loose connections

- NuPRL.
- PX: A computational logic.
- Smith, "An interpretation of Martin-Löf's type theory in a type-free theory of propositions", JSL, 49, 1984.
- Bunder, Dekkers, "Pure Type Systems with More Liberal Rules", JSL, 66, 2001.
- Logical frameworks.

Technical results

1. Conservativity of a classical higher-order illative system over standard semantics for higher-order logic.
2. Consistency of classical higher-order illative systems with dependent types, predicate subtypes, W-types, and:
   - Non-constructive choice.
   - Conditional (branching on truth-values).
   - Extensionality for W-types.

Technical results

Model construction

A model is a tuple $\langle \mathcal{C}, \mathcal{T}, \mathcal{F}, \bot, \mathsf{v}, \forall, \ldots \rangle$ where :

1. $\bot \in \mathcal{F}$,
2. $\mathcal{T} \cap \mathcal{F} = \emptyset$.
3. $\mathsf{v} \cdot a \cdot b \in \mathcal{T}$ iff $a \in \mathcal{T}$ or $b \in \mathcal{T}$,
4. $\mathsf{v} \cdot a \cdot b \in \mathcal{F}$ iff $a \in \mathcal{F}$ and $b \in \mathcal{F}$,
5. $\forall \cdot a \cdot b \in \mathcal{T}$ iff $\mathrm{Type} \cdot a \in \mathcal{T}$ and for every $c \in \mathcal{C}$ with $a \cdot c \in \mathcal{C}$ we have $b \cdot c \in \mathcal{C}$,
6. $\forall \cdot a \cdot b \in \mathcal{F}$ iff $\mathrm{Type} \cdot a \in \mathcal{T}$ and there exists $c \in \mathcal{C}$ with $a \cdot c \in \mathcal{T}$ and $b \cdot c \in \mathcal{F}$,
7. if $\mathrm{Type} \cdot a \in \mathcal{T}$ and $\mathrm{Type} \cdot b \in \mathcal{T}$, then $\mathrm{Type} \cdot (\rightarrow \cdot a \cdot b) \in \mathcal{T}$,
8. $\ldots$

Technical results

Model construction

We construct a term model parameterised by a standard model $\mathcal{N}$ for higher-order logic.

$$\mathcal{N} = \langle \{\mathcal{D}_\tau \mid \tau \in \mathscr{T}\}, I \rangle$$

We construct a term model parameterised by a standard model $\mathcal{N}$ for higher-order logic.

$$\mathcal{N} = \langle \{\mathcal{D}_\tau \mid \tau \in \mathscr{T}\}, I \rangle$$

Types of higher-order logic:

$$\mathscr{T} \quad ::= \quad o \mid \iota \mid \mathscr{T} \to \mathscr{T}$$

Technical results

Model construction

We construct a term model parameterised by a standard model $\mathcal{N}$ for higher-order logic.

$$\mathcal{N} = \langle \{\mathcal{D}_\tau \mid \tau \in \mathscr{T}\}, I \rangle$$

Types of higher-order logic:

$$\mathscr{T} \quad ::= \quad o \mid \iota \mid \mathscr{T} \to \mathscr{T}$$

For a type $\tau \in \mathscr{T}$ and an ordinal α we define the representation relations $\succ_\tau^\alpha \in \mathbb{T} \times \mathbb{T}$, the contraction relation $\to^\alpha \in \mathbb{T} \times \mathbb{T}$, and the relation $\succ_{\mathscr{T}}^\alpha \in \mathbb{T} \times \mathscr{T}$ inductively.

Technical results

Model construction

(β) $(\lambda x.X)Y \to^{\alpha} X[x/Y]$,

(γ) $fX \to^{\alpha} b$ if $f \in \mathcal{D}_{\tau_1 \to \tau_2}$, $a \in \mathcal{D}_{\tau_1}$, $b \in \mathcal{D}_{\tau_2}$, $f^{\mathcal{N}}(a) = b$ and $X \succ_{\tau_1}^{<\alpha} a$,

($\mathcal{F}_\tau$) $X \succ_{\tau}^{\alpha} d$ if $\tau = \tau_1 \to \tau_2$, $d \in \mathcal{D}_{\tau_1 \to \tau_2}$ and for every $a \in \mathcal{D}_{\tau_1}$ we have $Xa \rightsquigarrow_{\tau_2}^{<\alpha} d^{\mathcal{N}}(a)$,

($\mathrm{V}_\top$) $X \vee Y \succ_{o}^{\alpha} \top$ if $X \succ_{o}^{<\alpha} \top$ or $Y \succ_{o}^{<\alpha} \top$,

($\mathrm{V}_\bot$) $X \vee Y \succ_{o}^{\alpha} \bot$ if $X \succ_{o}^{<\alpha} \bot$ and $Y \succ_{o}^{<\alpha} \bot$,

($\forall_\top$) $\forall XY \succ_{o}^{\alpha} \top$ if $X \succ_{\mathscr{T}}^{<\alpha} \tau$ and for every $d \in \mathcal{D}_\tau$ we have $Yd \rightsquigarrow_{o}^{<\alpha} \top$,

($\forall_\bot$) $\forall XY \succ_{o}^{\alpha} \bot$ if $X \succ_{\mathscr{T}}^{<\alpha} \tau$ and there exists $d \in \mathcal{D}_\tau$ with $Yd \rightsquigarrow_{o}^{<\alpha} \bot$,

($\mathrm{F}_{\mathscr{T}}$) $X \to Y \succ_{\mathscr{T}}^{\alpha} \tau_1 \to \tau_2$ if $X \succ_{\mathscr{T}}^{<\alpha} \tau_1$ and $Y \succ_{\mathscr{T}}^{<\alpha} \tau_2$,

- ...

Technical results

Difficulty

Elements of type $\mathrm{Prop} \to \tau$, $(\alpha \to \mathrm{Prop}) \to \tau$, etc. for $\tau \neq \mathrm{Prop}$.