Formal Systems: Combinatory Logic and λ-calculus

Andrew R. Plummer

Department of Linguistics
The Ohio State University

30 Sept., 2009

Outline

COMBINATORY LOGIC

We present the foundations of Combinatory Logic and the λ-calculus. We mean to precisely demonstrate their similarities and differences.

CURRY AND FEYS (KOREAN FACE)

The material discussed is drawn from:

- Combinatory Logic Vol. 1, (1958) Curry and Feys.
- Lambda-Calculus and Combinators, (2008) Hindley and Seldin.

FORMAL SYSTEMS

We begin with some definitions.

FORMAL SYSTEMS

A *formal system* is composed of:

- A set of *terms*;
- A set of *statements* about terms;
- A set of statements, which are true, called *theorems*.

FORMAL SYSTEMS

TERMS

- We are given a set of *atomic terms*, which are unanalyzed primitives.
- We are also given a set of *operations*, each of which is a mode for combining a finite sequence of terms to form a new term.
- Finally, we are given a set of *term formation rules* detailing how to use the operations to form terms.

FORMAL SYSTEMS

STATEMENTS

- We are given a set of *predicates*, each of which is a mode for forming a statement from a finite sequence of terms.
- We are given a set of *statement formation rules* detailing how to use the predicates to form statements.

FORMAL SYSTEMS

THEOREMS

- We are given a set of *axioms*, each of which is a statement that is unconditionally true (and thus a theorem).
- We are given a set of *deductive rules* detailing how to use the axioms to derive other theorems.

ELEMENTARY THEORY OF NUMERALS

We need to specify: a set of atomic terms, a set of operations, a set of term formation rules.

TERMS

- Let $\{0\}$ be the set of atomic terms.
- Let $\{S\}$ be the set of operations.
- Let $\{\textsc{Suc}\}$ be the set of term formation rules where:

  SUC: If x is a term, then Sx is also a term.

ELEMENTARY THEORY OF NUMERALS

We need to specify a set of predicates, and a set of statement formation rules.

STATEMENTS

- Let $\{=\}$ be the set of binary predicates.
- Let $\{\text{EQ}\}$ be the set statement formation rules where:

  EQ: If x and y are terms, then $x = y$ is a statement.

ELEMENTARY THEORY OF NUMERALS

We need to specify a set of axioms, and a set of deductive rules.

THEOREMS

- Let $\{0 = 0\}$ be the set of axioms.
- Let $\{\text{EQOFSUC}\}$ be the set of deductive rules where:

$$\text{EQOFSUC: If } x = y \text{, then } Sx = Sy.$$

ELEMENTARY THEORY OF NUMERALS

DERIVATIONS OF TERMS

$$\frac{0}{S0}\,\textsc{Suc}$$

$$\frac{SS0}{SSS0}\,\textsc{Suc}$$

ELEMENTARY THEORY OF NUMERALS

DERIVATIONS OF STATEMENTS

$$\frac{0 \qquad 0}{0 = 0}\ \textsc{Eq}$$

$$\frac{S0 \qquad 0}{S0 = 0}\ \textsc{Eq}$$

ELEMENTARY THEORY OF NUMERALS

DERIVATIONS OF THEOREMS

$$\dfrac{0 = 0}{S0 = S0}\ \textsc{EqOfSuc}$$

$$\dfrac{\dfrac{0 = 0}{S0 = S0}\ \textsc{EqOfSuc}}{SS0 = SS0}\ \textsc{EqOfSuc}$$

Outline

Applicative Systems

Let $\mathcal{F}$ be a formal system.
Let $\mathcal{T}$ be the set of terms of $\mathcal{F}$.
Let $\mathcal{O} = \{f_1, f_2, \ldots, f_n\}$ be the set of operations of $\mathcal{F}$.

Schönfinkel's Reduction Algorithm

- Given n-ary $f_i \in \mathcal{O}$, add a fresh F_i to $\mathcal{T}$.
- Add a new binary operation app (called *application*) to $\mathcal{O}$, and denote it by juxtaposition, which is left associative.
- Define the term $f(t_1, \ldots, t_n)$ as $F_i t_1 \cdots t_n$.
- Remove f_i from $\mathcal{O}$.
- Return a formal system $\mathcal{F}^i_{\mathrm{app}}$ just like $\mathcal{F}$, only less f_i, and with app.

APPLICATIVE SYSTEMS

EXAMPLE

- Suppose a formal system has the binary operation $+$ and the term formation rule

$$\frac{x \qquad y}{x + y}\ \textsc{Plus}$$

- We adjoin the fresh *add* to the set of terms, and define $x + y$ as *add* x y (as seen in haskell!).

APPLICATIVE SYSTEMS

QUASI-APPLICATIVE SYSTEMS

A *quasi-applicative system* is a formal system whose set of operations contains application.

By iteration of Schönfinkel's Reduction Algorithm we can remove as many operations (other than app) as we want.

APPLICATIVE SYSTEMS

An *applicative system* is a formal system whose only operation is application.

We can reduce any formal system to an applicative system.

FUNCTIONAL ABSTRACTION

FUNCTIONAL ABSTRACTION

Functional Abstraction (or simply *abstraction*) is a binary operation, designated by a prefixed 'λ', with the following term formation rule:

ABS: If x is a variable[1] and M a term, then $\lambda x M$ is also a term.

λ-APPLICATIVE SYSTEMS

A *λ-applicative system* is a formal system whose only operations are application and abstraction.

[1] We will not formally define variables. We simply assume that they are particular members of the set of atomic terms.

COMBINATIONS

COMBINATIONS

A *combination* is a term formed by utilizing application zero or more times.

COMBINATORIAL COMPLETENESS

A formal system is *combinatorially complete* if any function we can define intuitively by means of a variable can be represented by a combination yielded by the system.

COMBINATORS

COMBINATORS

A *combinator* is a kind of combination. We want to define them such that they provide combinatorial completeness.

CROSSROADS

We have two viable options:

- Define combinators via abstraction. This leads to the formulation of the λ-calculus as a special class of λ-applicative systems.
- Postulate certain combinators as atomic terms, and define all others using them. This leads to the theory of combinators as a special class of applicative systems.

THEORY OF COMBINATORS

Given that we are already familiar with the λ-calculus, we will develop the theory of combinators as applicative systems.

After developing the applicative theory, we will demonstrate a transform between applicative systems and λ-applicative systems.

THEORY OF COMBINATORS

We need to specify: a set of atomic terms, a set of operations, a set of term formation rules.

TERMS

- Let $\{\mathbf{S}, \mathbf{K}, \mathbf{I}, x_1, x_2, \ldots, f_1, f_2, \ldots\}$ be the set of atomic terms.
- Let $\{\text{app}\}$ be the set of operations.
- Let $\{\text{APP}\}$ be the set of term formation rules where:

  APP: If x and y are terms, then xy is also a term.

A *combinator* is any term built from **S**, **K**, or **I** by zero or more uses of app; e.g. **SKK**.

Theory of Combinators

We need to specify a set of predicates, and a set of statement formation rules.

Statements

- Let $\{\triangleright\}$ be the set of binary predicates.
- Let $\{\textsc{Red}\}$ be the set statement formation rules where:

  Red: If x and y are terms, then $x \triangleright y$ is a statement.

$\triangleright$ is called reduction, and will constitute a monotonic partial order on terms.

THEORY OF COMBINATORS

We need to specify a set of axioms, and a set of deductive rules.

AXIOMS

Let x, y, z be terms. The set of axioms contains the following statement schemas:

- $\mathbf{I}x = x$
- $\mathbf{K}xy = x$
- $\mathbf{S}xyz = xz(yz)$

THEORY OF COMBINATORS

Let x, y, z be terms.

DEDUCTIVE RULES

Let $\{$REF, TRANS, RMON, LMON$\}$ be the set of deductive rules where:

- REF: $x \triangleright x$,
- TRANS: If $x \triangleright y$ and $y \triangleright z$, them $x \triangleright z$,
- RMON: If $x \triangleright y$ them $zx \triangleright zy$,
- LMON: If $x \triangleright y$ them $xz \triangleright yz$.

We can include a symmetry rule SYM to make $\triangleright$ into a monotonic equivalence relation $=$.

EXAMPLE

DERIVATION OF $(((\mathbf{S}(\mathbf{KS})\mathbf{K})g)f)x \rhd g(fx)$

$$\dfrac{(((\mathbf{S}(\mathbf{KS})\mathbf{K})g)f)x \rhd ((((\mathbf{KS})g)(\mathbf{K}g))f)x \qquad ((((\mathbf{KS})g)(\mathbf{K}g))f)x \rhd ((\mathbf{S}(\mathbf{K}g))f)x}{(1)\ (((\mathbf{S}(\mathbf{KS})\mathbf{K})g)f)x \rhd ((\mathbf{S}(\mathbf{K}g))f)x}\ \textsc{Trans}$$

$$\dfrac{((\mathbf{S}(\mathbf{K}g))f)x \rhd ((\mathbf{K}g)f)(fx) \qquad ((\mathbf{K}g)f)(fx) \rhd g(fx)}{(2)\ ((\mathbf{S}(\mathbf{K}g))f)x \rhd g(fx)}\ \textsc{Trans}$$

$$\dfrac{(1)\ (((\mathbf{S}(\mathbf{KS})\mathbf{K})g)f)x \rhd ((\mathbf{S}(\mathbf{K}g))f)x \qquad (2)\ ((\mathbf{S}(\mathbf{K}g))f)x \rhd g(fx)}{(((\mathbf{S}(\mathbf{KS})\mathbf{K})g)f)x \rhd g(fx)}\ \textsc{Trans}$$

EXAMPLE

DERIVATION OF $(((\mathbf{S}(\mathbf{KS})\mathbf{K})g)f)x \rhd g(fx)$

$(((\mathbf{S}(\mathbf{KS})\mathbf{K})g)f)x$	$\rhd$	$((((\mathbf{KS})g)(\mathbf{K}g))f)x$	$(\mathbf{S}(\mathbf{KS})\mathbf{K})g \to ((\mathbf{KS})g)(\mathbf{K}g)$
	$\rhd$	$((\mathbf{S}(\mathbf{K}g))f)x$	$(\mathbf{KS})g \to \mathbf{S}$
	$\rhd$	$((\mathbf{K}g)f)(fx)$	Reducing $((\mathbf{S}(\mathbf{K}g))f)x$
	$\rhd$	$g(fx)$	$((\mathbf{K}g)f) \to g$

We define the combinator **B** as **S**(**KS**)**K**. Assuming the f and g are functions, it is easy to see that **B** is function composition.

COMBINATORIAL COMPLETENESS

We still need to show that the theory of combinators is combinatorial complete.

We know that the λ-calculus is combinatorially complete. Thus it is enough to show that the theory of combinators is equivalent to λ-calculus.

That is, given any λ-term *M*, we show that **S**, **K** and **I** can be composed to produce a combinator equivalent to *M*, and vice versa.

COMBINATORS TO λ-CALCULUS

LET LC[$\cdot$] BE THE FOLLOWING TRANSFORMATION

- LC[x] = x (x a variable)
- LC[**I**] $= \lambda x.x$
- LC[**K**] $= \lambda x.\lambda y.x$
- LC[**S**] $= \lambda x.\lambda y.\lambda z.(xz(yz))$
- LC[($M_1 M_2$)] = (LC[M_1] LC[M_2])

λ-CALCULUS TO COMBINATORS

LET CL[·] BE THE FOLLOWING TRANSFORMATION

- $\mathrm{CL}[x] = x$ (x a variable)
- $\mathrm{CL}[(M_1 M_2)] = (\mathrm{CL}[M_1]\ \mathrm{CL}[M_2])$
- $\mathrm{CL}[\lambda x.M] = (\mathbf{K}\ \mathrm{CL}[M])$ (if x is not free in M)
- $\mathrm{CL}[\lambda x.x] = \mathbf{I}$
- $\mathrm{CL}[\lambda x.\lambda y.M] = \mathrm{CL}[\lambda x.\ \mathrm{CL}[\lambda y.M]]$ (if x is free in M)
- $\mathrm{CL}[\lambda x.(M_1 M_2)] = (\mathbf{S}\ \mathrm{CL}[\lambda x.M_1]\ \mathrm{CL}[\lambda x.M_2])$

Outline

1. Introduction
2. Applicative systems
3. Useful Information

NORMAL FORMS

COMBINATORY NORMAL FORM

Let x, y, z be terms. A term $\mathbf{I}x$, $\mathbf{K}xy$, or $\mathbf{S}xyz$ is called a *combinatory redex*. A *combinatory normal form* is a combinatory term that contains no combinatory redexes.

β-NORMAL FORM

Let M and N be λ-terms. A term $(\lambda xM)N$ is called a β*-redex*. A β*-normal form* is a λ-term that contains no β-redexes.

CHURCH-ROSSER

CHURCH-ROSSER THEOREM FOR $\triangleright$

Let x, y, t be combinatory terms. If $t \triangleright x$ and $t \triangleright y$, then there exists a combinatory term z such that $x \triangleright z$ and $y \triangleright z$.

CHURCH-ROSSER THEOREM FOR $\triangleright_\beta$[2]

Let M, N, T be λ-terms. If $T \triangleright_\beta M$ and $T \triangleright_\beta N$, then there exists a term P such that $M \triangleright_\beta P$ and $N \triangleright_\beta P$.

---

[2] This is just β-reduction.

UNIQUENESS OF NORMAL FORMS

UNIQUENESS OF COMBINATORY NORMAL FORMS

A combinatory term can have at most one *combinatory normal form*.

UNIQUENESS OF β-NORMAL FORMS

A λ-term can have at most one β*-normal form*.

FIXED-POINT COMBINATOR

We briefly discuss the *fixed-point combinator* $\mathbf{Y}$.

Let f be a term and consider the term $\mathbf{Y}f$. Since this combinator is to be thought of as implementing recursion, we need $\mathbf{Y}f$ to be a term that has both f and $\mathbf{Y}$ in it somehow. A natural choice is

$$\mathbf{Y}f = f(\mathbf{Y}f).$$

That is, we just evaluate f at $\mathbf{Y}f$.

FIXED-POINT COMBINATOR

To justify our definition of **Y**, we consider its formulation in the λ-calculus, due to Curry.

$$\mathbf{Y} = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

YG = G(**Y**G)

$\mathbf{Y}g =$	$(\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))g$	
$\mathbf{Y}g =$	$(\lambda x.g(xx))(\lambda x.g(xx))$	β-reduction
$\mathbf{Y}g =$	$(\lambda y.g(yy))(\lambda x.g(xx))$	α-conversion
$\mathbf{Y}g =$	$g((\lambda x.g(xx))(\lambda x.g(xx)))$	β-reduction
$\mathbf{Y}g =$	$g(\mathbf{Y}g)$	from second line

SK-BASIS FOR COMBINATORY LOGIC

Any combinator can be defined using only the combinators **S** and **K**.

We can define **I** as **SKK**.

Similarly, we can define the fixed point combinator as

$$\mathbf{Y} = \mathbf{SSK(S(K(SS(S(SSK))))K)}$$