λ-Calculus: Recursion Operators and Types

Dana S. Scott

University Professor, Emeritus
Carnegie Mellon University
Visiting Scholar in Mathematics
University of California, Berkeley

May, 2014
Church's $\lambda$-Calculus

**Definition.** $\lambda$-calculus — as a formal theory — has rules for the *explicit definition* of functions *via* equational axioms:

**$\alpha$-conversion**
\[
\lambda x.[\ldots x\ldots] = \lambda y.[\ldots y\ldots]
\]

**$\beta$-conversion**
\[
(\lambda x.[\ldots x\ldots])(t) = [\ldots t\ldots]
\]

**$\eta$-conversion**
\[
\lambda x. f(x) = f
\]

The basic syntax has one binary operation of *application* and one variable-binding operator of *abstraction*. These are the "logical" notions of the theory, but we can add other *constants* for special operators.

Note that third axiom will be dropped in favor of a theory employing properties of a partial ordering.
The Graph Model

Definitions. (1). Pairing: \((n, m) = 2^n (2m + 1)\).

(2). Sequence numbers: \(\langle \rangle = 0\) and
\[\langle n_0, n_1, \ldots, n_{k-1}, n_k \rangle = (\langle n_0, n_1, \ldots, n_{k-1} \rangle, n_k)\].

(3). Sets: \(\text{set}(0) = \emptyset\) and \(\text{set}((n, m)) = \text{set}(n) \cup \{m\}\).

(4). Kleene star: \(X^* = \{ n \mid \text{set}(n) \subseteq X \}\), for sets \(X \subseteq \mathbb{N}\).

Definition. The **enumeration operator model** is given by these definitions on **sets** of integers:

**Application**

\[F(X) = \{ m \mid \exists n \in X^*. (n, m) \in F \}\]

**Abstraction**

\[
\lambda x. [\ldots x \ldots] = \\
\{ 0 \} \cup \{(n, m) \mid m \in [\ldots \text{set}(n) \ldots] \}
\]

NOTE: This model could easily have been defined in 1957, and it satisfies the rules of \(\alpha, \beta\)-conversion (but not \(\eta\)).

(Some historical comments can be found at the end of these notes.)
What is the Secret?

(1) The powerset \( P(\mathbb{N}) = \{ x \mid x \subseteq \mathbb{N} \} \) is a topological space with the sets \( U_n = \{ x \mid n \in x^* \} \) as a basis for the topology—the positive topology.

(2) Functions \( \Phi : P(\mathbb{N})^n \to P(\mathbb{N}) \) are continuous iff, for all integers, \( m \in \Phi(\{ x_0, x_1, \ldots, x_{n-1} \}) \) iff there are \( k_i \in x_i^* \) for all \( i < n \), such that \( m \in \Phi(\text{set}(k_0), \ldots, \text{set}(k_{n-1})) \).

(3) The application operation \( F(x) \) is continuous as a function of two variables.

(4) If \( \Phi(\{ x_0, x_1, \ldots, x_{n-1} \}) \) is continuous, then the abstraction \( \lambda x_0. \Phi(\{ x_0, x_1, \ldots, x_{n-1} \}) \) is continuous in all of the remaining variables.

(5) If \( \Phi(x) \) is continuous, then \( \lambda x. \Phi(x) \) is the largest set \( F \) such that for all sets \( T \), we have \( F(T) = \Phi(T) \).

(6) And, note, therefore, that generally \( F \subseteq \lambda x. F(x) \).
Some Lambda Properties

For all sets of integers \( F \) and \( G \) we have:

\[
\lambda x. F(x) \subseteq \lambda x. G(x) \iff \forall x. F(x) \subseteq G(x),
\]

\[
\lambda x. (F(x) \cap G(x)) = \lambda x. F(x) \cap \lambda x. G(x),
\]

and

\[
\lambda x. (F(x) \cup G(x)) = \lambda x. F(x) \cup \lambda x. G(x).
\]

Definition. A continuous operator \( \Phi(x_0, x_1, \ldots, x_{n-1}) \) is \textit{computable} iff in the model this set is RE:

\[
F = \lambda x_0 \lambda x_1 \ldots \lambda x_{n-1}. \Phi(x_0, x_1, \ldots, x_{n-1}).
\]

Theorems.
- All pure \( \lambda \)-terms define \textit{computable} operators.
- If \( \Phi(x) \) is continuous and we let \( \nabla = \lambda x. \Phi(x(x)) \), then \( P = \nabla(\nabla) \) is the \textit{least fixed point} of \( \Phi \).
- The least fixed point of a \textit{computable} operator is always computable.

\[
\text{Succ}(X) = \{n+1 \mid n \in X\}, \quad \text{Pred}(X) = \{n \mid n+1 \in X\}, \quad \text{and}
\]

\[
\text{Test}(Z)(X)(Y) = \{n \in X \mid 0 \in Z\} \cup \{m \in Y \mid \exists k. k+1 \in Z\},
\]

with \( \lambda \)-calculus, suffice for defining all RE sets.
Gödel Numbering

**Lemma.** There is a computable $V = \lambda x. V(x)$ where

(i) $V(\{0\}) = \lambda y. \lambda x. y$,
(ii) $V(\{1\}) = \lambda z. \lambda y. \lambda x. z(x)(y(x))$,
(iii) $V(\{2\}) = \text{Test}$,
(iv) $V(\{3\}) = \text{Succ}$,
(v) $V(\{4\}) = \text{Pred}$, and
(vi) $V(\{4+(n,m)\}) = V(\{n\})(V(\{m\}))$.

**Theorem.** Every *recursively enumerable set* is of the form $V(\{n\})$.

**Definition.** Modify the definition of $V$ via *finite approximations*:

(i) $V_k(\{n\}) = V(\{n\}) \cap \{i \mid i < k\}$ for $n < 5$, and
(ii) $V_k(\{4+(n,m)\}) = V_k(\{n\})(V_k(\{m\}))$.

**Theorem.** Each $V_k(\{n\}) \subseteq V_{k+1}(\{n\})$ is *finite*, the predicate $j \in V_k(\{n\})$ is *recursive*, and we have:

$$V(\{n\}) = \bigcup_{k < \infty} V_k(\{n\}).$$

**Theorem.** The sets $\mathcal{L}_0$ and $\mathcal{L}_1$ are *recursively enumerable*, *disjoint*, and *recursively inseparable*:

$\mathcal{L}_0 = \{n \mid \exists j \ [0 \in V_j(\{n\})(\{n\}) \land 1 \notin V_j(\{n\})(\{n\})]\}$

$\mathcal{L}_1 = \{n \mid \exists k \ [1 \in V_k(\{n\})(\{n\}) \land 0 \notin V_k(\{n\})(\{n\})]\}$
What is a Type?

Definition. Using pairing functions we may regard \( \mathcal{P}(\mathbb{N}) = \mathcal{P}(\mathbb{N}) \times \mathcal{P}(\mathbb{N}) \), and for \( \mathcal{A} \subseteq \mathcal{P}(\mathbb{N}) \) we write
\[
x \mathcal{A} y \text{ iff } (x, y) \in \mathcal{A}.
\]

Definition. By a type over \( \mathcal{P}(\mathbb{N}) \) we understand a partial equivalence relation \( \mathcal{A} \subseteq \mathcal{P}(\mathbb{N}) \) where,
\[
\text{for all } x, y, z \in \mathcal{P}(\mathbb{N}), \text{ we have}
\]
\[
x \mathcal{A} y \text{ implies } y \mathcal{A} x, \text{ and}
\]
\[
x \mathcal{A} y \text{ and } y \mathcal{A} z \text{ imply } x \mathcal{A} z.
\]
Additionally we write \( x:\mathcal{A} \text{ iff } x \mathcal{A} x \).

Note: It is better NOT to pass to equivalence classes and the corresponding quotient spaces.

But we can THINK in those terms if we like, as this is a very common construction.

Definition. For subspaces \( \mathcal{X} \subseteq \mathcal{P}(\mathbb{N}) \), write
\[
[\mathcal{X}] = \{ (x, x) \mid x \in \mathcal{X} \},
\]
so that we may regard \textit{subspaces as types.}
The Category of Types

**Definition.** The *product* of types $\mathcal{A}, \mathcal{B} \subseteq \mathcal{P}(\mathbb{N})$ is defined as that relation where

$$X(\mathcal{A} \times \mathcal{B})Y \iff \text{Fst}(X) : \mathcal{A} \text{ and } \text{Snd}(X) : \mathcal{B}.$$ 

**Theorem.** The product of two types is again a type, and we have

$$X : (\mathcal{A} \times \mathcal{B}) \iff \text{Fst}(X) : \mathcal{A} \text{ and } \text{Snd}(X) : \mathcal{B}.$$ 

**Definition.** The *exponentiation* of types $\mathcal{A}, \mathcal{B} \subseteq \mathcal{P}(\mathbb{N})$ is defined as that relation where

$$F(\mathcal{A} \to \mathcal{B})G \iff \forall X, Y. \ X \mathcal{A} Y \implies F(X) \mathcal{B} G(Y).$$ 

**Theorem.** The exponentiation (= function space) of two types is again a type, and we have

$$F : \mathcal{A} \to \mathcal{B} \implies \forall X. \ X : \mathcal{A} \implies F(X) : \mathcal{B}.$$ 

**Note:** Types do form a category — expanding the topological category of subspaces — but we wish to prove much, much more.
Isomorphism of Types

Definition. The sum of types $\mathcal{A}, \mathcal{B} \subseteq \mathcal{P}(\mathbb{N})$ is defined as that relation where $X(\mathcal{A} + \mathcal{B})Y$ iff either $\exists X_0, Y_0 [X_0 \mathcal{A} Y_0 \& X = (0, X_0) \& Y = (0, Y_0)]$ or $\exists X_1, Y_1 [X_1 \mathcal{B} Y_1 \& X = (1, X_1) \& Y = (1, Y_1)]$.

Theorem. The sum of two types is again a type, and we have $X : (\mathcal{A} + \mathcal{B})$ iff either $\text{Fst}(X) = 0$ & $\text{Snd}(X) : \mathcal{A}$ or $\text{Fst}(X) = 1$ & $\text{Snd}(X) : \mathcal{B}$.

Definition. Two types $\mathcal{A}, \mathcal{B} \subseteq \mathcal{P}(\mathbb{N})$ are isomorphic, in symbols $\mathcal{A} \cong \mathcal{B}$, provided there are $F : \mathcal{A} \rightarrow \mathcal{B}$ and $G : \mathcal{B} \rightarrow \mathcal{A}$ where $\forall X : \mathcal{A}. X \mathcal{A} G(F(X))$ and $\forall Y : \mathcal{B}. Y \mathcal{B} F(G(Y))$.

Theorem. If types $\mathcal{A}_0 \cong \mathcal{B}_0$ and $\mathcal{A}_1 \cong \mathcal{B}_1$, then $(\mathcal{A}_0 \times \mathcal{A}_1) \cong (\mathcal{B}_0 \times \mathcal{B}_1)$, and $(\mathcal{A}_0 + \mathcal{A}_1) \cong (\mathcal{B}_0 + \mathcal{B}_1)$, and $(\mathcal{A}_0 \rightarrow \mathcal{A}_1) \cong (\mathcal{B}_0 \rightarrow \mathcal{B}_1)$. 
Dependent Types

Definition. Let \( \mathcal{T}' \) be the class of all types on the powerset space \( \mathcal{P}(\mathbb{N}) \). For \( A \in \mathcal{T}' \), an \( A \)-indexed family of types is a function \( \beta : \mathcal{P}(\mathbb{N}) \to \mathcal{T}' \), such that

\[
\forall x_0, x_1. \ x_0 A x_1 \implies \beta(x_0) = \beta(x_1).
\]

Definition. The dependent product of an \( A \)-indexed family of types, \( \beta \), is defined as that relation such that

\[
F_0(\prod x : A. \beta(x)) F_1 \iff
\forall x_0, x_1. \ x_0 A x_1 \implies F_0(x_0) \beta(x_0) F_1(x_1).
\]

Definition. The dependent sum of an \( A \)-indexed family of types, \( \beta \), is defined as that relation such that

\[
z_0(\sum x : A. \beta(x)) z_1 \iff
\exists x_0, y_0, x_1, y_1 [ x_0 A x_1 & y_0 \beta(x_0) y_1 \&
\]

\[
z_0 = (x_0, y_0) \& z_1 = (x_1, y_1) \]

Theorem. The dependent products and dependent sums of indexed families of types are again types.
Systems of Dependent Types

**Definition.** We say that $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$ form a system of dependent types iff

- $\forall x_0, x_1. [x_0 \mathcal{A} x_1 \Rightarrow \mathcal{B}(x_0) = \mathcal{B}(x_1)]$, and
- $\forall x_0, x_1, y_0, y_1. [x_0 \mathcal{A} x_1 \& y_0 \mathcal{B}(x_0) y_1 \Rightarrow \mathcal{C}(x_0, y_0) = \mathcal{C}(x_1, y_1)]$, and
- $\forall x_0, x_1, y_0, y_1, z_0, z_1. [x_0 \mathcal{A} x_1 \& y_0 \mathcal{B}(x_0) y_1 \& z_0 \mathcal{C}(x_0, y_0) z_1 \Rightarrow \mathcal{D}(x_0, y_0, z_0) = \mathcal{D}(x_1, y_1, z_1)]$,

provided that $\mathcal{A} \in \mathcal{T}$, and $\mathcal{B}, \mathcal{C}, \mathcal{D}$ are functions on $\mathcal{P}(\mathbb{N})$ to $\mathcal{T}'$ of the indicated number of arguments.

**Note:** Clearly the definition can be extended to systems of any number of terms.

**Theorem.** Under the above assumptions on $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$, we always have

$$\prod x: \mathcal{A}. \sum y: \mathcal{B}(x). \prod z: \mathcal{C}(x, y). \mathcal{D}(x, y, z) \in \mathcal{T}'$$
Asserting Propositions

Definition. Every type \( \mathcal{P} \in \mathcal{T} \) can be regarded as a proposition where asserting (or proving \( \mathcal{P} \)) means finding evidence \( \mathcal{E} : \mathcal{P} \).

Note: Under this interpretation of logic, asserting \( (\mathcal{P} \times \mathcal{Q}) \) means asserting a conjunction, asserting \( (\mathcal{P} + \mathcal{Q}) \) means asserting a disjunction, asserting \( (\mathcal{P} \rightarrow \mathcal{Q}) \) means asserting an implication, asserting \( (\prod_{x : \mathcal{A}} \mathcal{P}(x)) \) means asserting a universal quantification, and asserting \( (\sum_{x : \mathcal{A}} \mathcal{B}(x)) \) means asserting an existential quantification.

Definition. For \( \mathcal{A} \in \mathcal{T} \) the identity type on \( \mathcal{A} \) is defined as that relation such that

\[ Z(x \equiv_{\mathcal{A}} y) \text{ iff } Z \mathcal{A} x \mathcal{A} y \mathcal{A} w. \]

Example: Given \( \mathcal{F} : (\mathcal{A} \rightarrow (\mathcal{A} \rightarrow \mathcal{A})) \), then asserting

\[ \prod_{x : \mathcal{A}} \prod_{y : \mathcal{A}} \prod_{z : \mathcal{A}} \mathcal{F}(x)(\mathcal{F}(y)(z)) \equiv_{\mathcal{A}} \mathcal{F}(\mathcal{F}(x)(y))(z) \]

means asserting that \( \mathcal{F} \) is an associative operation.
Some Background References

There are many approaches to modeling $\lambda$-calculus, and expositions and historical references can be found in Cardone-Hindley [2009]. In 1972 Plotkin wrote an AI report at the University of Edinburgh entitled "A set-theoretical definition of application" which remained unpublished until it was incorporated into the more extensive paper Plotkin [1993], which discusses many kinds of models. Scott developed his model based on the powerset of the integers subsequently, but he only later realized it was basically the same as Plotkin's model. See Scott [1976] for further details where he called the idea The Graph Model.


Much earlier, enumeration reducibility was introduced by Rogers in lecture notes and mentioned by Friedberg-Rogers [1959] as a way of defining a positive reducibility between sets. Enumeration degrees are discussed at length in Rogers [1967]. There is now a vast literature on the subject. Enumeration operators are also studied in Rogers [1967] as well. Earlier, Myhill-Shepherdson [1955] defined functionals on partial functions with similar properties. Neither team saw that their operators possessed an algebra that would model $\lambda$-calculus, however.

More Background References

Some historical remarks on the notion of partial equivalence relations (PERs) as an interpretation of types are given by Bruce et al. [1990], where we learn that they were introduced by Myhill and Shepherdson [1955] for types of first-order functions, and then extended to simple types by Kreisel [1959]. Scott took the use of partial equivalence relations from the work of Kreisel and collaborators. More recent material and references can be found in the books by Gunter and Mitchell [1994] and Mitchell [1996]. An influential paper to consult is Abadi and Plotkin [1990]


