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 λ-Calculus

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

α-conversion

λ`X.[...X...] = `λ`Y.[...Y...]`

β-conversion

`(`λ`X.[...X...])(T) = [...T...]`

η-conversion

λ`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:*** $\mathbf{set}(0) = \varnothing$ and $\mathbf{set}((n,m)) = \mathbf{set}(n) \cup \{m\}$.

(4). ***Kleene star:*** $X^* = \{n \mid \mathbf{set}(n) \subseteq X\}$, 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

$$\lambda X.[\ldots X \ldots] =$$

$$\{0\} \cup \{(n,m) \mid m \in [\ldots \mathbf{set}(n) \ldots]\}$$

NOTE: This model could easily have been defined in 1957, and it satisfies the rules of α, β-conversion (but not η).

(Some historical comments can be found at the end of these notes.)

What is the Secret?

(1) The powerset $\mathcal{P}(\mathbb{N}) = \{\mathtt{X} \mid \mathtt{X} \subseteq \mathbb{N}\}$ is a ***topological space*** with the sets $\mathcal{U}_\mathtt{n} = \{\mathtt{X} \mid \mathtt{n} \in \mathtt{X}^*\}$ as a ***basis*** for the topology.

(2) Functions $\Phi : \mathcal{P}(\mathbb{N})^\mathtt{n} \longrightarrow \mathcal{P}(\mathbb{N})$ are ***continuous*** iff, for all integers, $\mathtt{m} \in \Phi(\mathtt{X}_0, \mathtt{X}_1, \ldots, \mathtt{X}_{\mathtt{n}-1})$ iff there are $\mathtt{k}_\mathtt{i} \in \mathtt{X}_\mathtt{i}^*$ for all $\mathtt{i} < \mathtt{n}$, such that $\mathtt{m} \in \Phi(\mathbf{set}(\mathtt{k}_0), \ldots, \mathbf{set}(\mathtt{k}_{\mathtt{n}-1}))$.

(3) The application operation $\mathtt{F}(\mathtt{X})$ is continuous as a function of ***two*** variables.

(4) If $\Phi(\mathtt{X}_0, \mathtt{X}_1, \ldots, \mathtt{X}_{\mathtt{n}-1})$ is continuous, then the abstraction $\boldsymbol{\lambda} \mathtt{X}_0 . \Phi(\mathtt{X}_0, \mathtt{X}_1, \ldots, \mathtt{X}_{\mathtt{n}-1})$ is continuous in all of the ***remaining variables***.

(5) If $\Phi(\mathtt{X})$ is continuous, then $\boldsymbol{\lambda} \mathtt{X} . \Phi(\mathtt{X})$ is the ***largest set*** $\mathtt{F}$ such that for all sets $\mathtt{T}$, we have $\mathtt{F}(\mathtt{T}) = \Phi(\mathtt{T})$.

(6) And, note, therefore, that generally $\mathtt{F} \subseteq \boldsymbol{\lambda} \mathtt{X} . \mathtt{F}(\mathtt{X})$.

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 ***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 λ-terms define ***computable*** operators.
- If $\Phi(X)$ is continuous and we let $\nabla = \lambda X.\Phi(X(X))$, then $P = \nabla(\nabla)$ is the ***least fixed point*** of Φ.
- The least fixed point of a ***computable*** operator is always computable.

Succ$(X) = \{n+1 \mid n \in X\}$, **Pred**$(X) = \{n \mid n+1 \in X\}$, **and**

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

with λ-calculus, suffice for defining all RE sets.

Gödel Numbering

Lemma. There is a computable $\mathbf{V} = \lambda \mathtt{X}.\, \mathbf{V}(\mathtt{X})$ where

(i) $\mathbf{V}(\{0\}) = \lambda \mathtt{Y}.\lambda \mathtt{X}.\,\mathtt{Y}$,

(ii) $\mathbf{V}(\{1\}) = \lambda \mathtt{Z}.\lambda \mathtt{Y}.\lambda \mathtt{X}.\,\mathtt{Z}(\mathtt{X})(\mathtt{Y}(\mathtt{X}))$,

(iii) $\mathbf{V}(\{2\}) = \mathbf{Test}$,

(iv) $\mathbf{V}(\{3\}) = \mathbf{Succ}$,

(v) $\mathbf{V}(\{4\}) = \mathbf{Pred}$, and

(vi) $\mathbf{V}(\{4+(\mathtt{n},\mathtt{m})\}) = \mathbf{V}(\{\mathtt{n}\})(\mathbf{V}(\{\mathtt{m}\}))$.

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

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

(i) $\mathbf{V}_\mathtt{k}(\{\mathtt{n}\}) = \mathbf{V}(\{\mathtt{n}\}) \cap \{\mathtt{i} \mid \mathtt{i} < \mathtt{k}\}$ for $\mathtt{n} < 5$, and

(ii) $\mathbf{V}_\mathtt{k}(\{4+(\mathtt{n},\mathtt{m})\}) = \mathbf{V}_\mathtt{k}(\{\mathtt{n}\})(\mathbf{V}_\mathtt{k}(\{\mathtt{m}\}))$.

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

$$\mathbf{V}(\{\mathtt{n}\}) = \bigcup_{\mathtt{k}<\infty} \mathbf{V}_\mathtt{k}(\{\mathtt{n}\}).$$

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

$$\mathcal{L}_0 = \{\mathtt{n} \mid \exists\, \mathtt{j}\, [0 \in \mathbf{V}_\mathtt{j}(\{\mathtt{n}\})(\{\mathtt{n}\}) \wedge 1 \notin \mathbf{V}_\mathtt{j}(\{\mathtt{n}\})(\{\mathtt{n}\})]\}$$

$$\mathcal{L}_1 = \{\mathtt{n} \mid \exists\, \mathtt{k}\, [1 \in \mathbf{V}_\mathtt{k}(\{\mathtt{n}\})(\{\mathtt{n}\}) \wedge 0 \notin \mathbf{V}_\mathtt{k}(\{\mathtt{n}\})(\{\mathtt{n}\})]\}$$

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

$\mathtt{X}\ \mathcal{A}\ \mathtt{Y}$ iff $(\mathtt{X},\mathtt{Y}) \in \mathcal{A}$.

Definition. By a ***type*** over $\mathcal{P}(\mathbb{N})$ we understand a ***partial equivalence relation*** $\mathcal{A} \subseteq \mathcal{P}(\mathbb{N})$ where, for all $\mathtt{X},\mathtt{Y},\mathtt{Z} \in \mathcal{P}(\mathbb{N})$, we have

$\mathtt{X}\ \mathcal{A}\ \mathtt{Y}$ implies $\mathtt{Y}\ \mathcal{A}\ \mathtt{X}$, and

$\mathtt{X}\ \mathcal{A}\ \mathtt{Y}$ and $\mathtt{Y}\ \mathcal{A}\ \mathtt{Z}$ imply $\mathtt{X}\ \mathcal{A}\ \mathtt{Z}$.

Additionally we write $\mathtt{X}:\mathcal{A}$ iff $\mathtt{X}\ \mathcal{A}\ \mathtt{X}$.

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 $\mathbb{X} \subseteq \mathcal{P}(\mathbb{N})$, write

$$[\mathbb{X}] = \{(\mathtt{X},\mathtt{X}) \mid \mathtt{X} \in \mathbb{X}\},$$

so that we may regard ***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

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

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

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

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

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

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

$$\mathtt{F}:\mathcal{A}\rightarrow\mathcal{B} \text{ implies } \forall \mathtt{X}.\ \mathtt{X}:\mathcal{A} \text{ implies } \mathtt{F}(\mathtt{X}):\mathcal{B}.$$

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 $\mathtt{X}(\mathcal{A}+\mathcal{B})\mathtt{Y}$ iff

either $\exists \mathtt{X}_0,\mathtt{Y}_0[\mathtt{X}_0\mathcal{A}\,\mathtt{Y}_0\ \&\ \mathtt{X}=(0,\mathtt{X}_0)\ \&\ \mathtt{Y}=(0,\mathtt{Y}_0)]$

or $\exists \mathtt{X}_1,\mathtt{Y}_1[\mathtt{X}_1\mathcal{B}\,\mathtt{Y}_1\ \&\ \mathtt{X}=(1,\mathtt{X}_1)\ \&\ \mathtt{Y}=(1,\mathtt{Y}_1)]$.

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

$\mathtt{X}:(\mathcal{A}+\mathcal{B})$ iff either $\mathbf{Fst}(\mathtt{X})=0\ \&\ \mathbf{Snd}(\mathtt{X}):\mathcal{A}$

or $\mathbf{Fst}(\mathtt{X})=1\ \&\ \mathbf{Snd}(\mathtt{X}):\mathcal{B}$.

Definition. Two types $\mathcal{A},\mathcal{B} \subseteq \mathcal{P}(\mathbb{N})$ are ***isomorphic***, in symbols $\mathcal{A}\cong\mathcal{B}$, provided there are

$\mathtt{F}:\mathcal{A}\to\mathcal{B}$ and $\mathtt{G}:\mathcal{B}\to\mathcal{A}$ where

$\forall \mathtt{X}:\mathcal{A}.\ \mathtt{X}\ \mathcal{A}\ \mathtt{G}(\mathtt{F}(\mathtt{X}))$ and $\forall \mathtt{Y}:\mathcal{B}.\ \mathtt{Y}\ \mathcal{B}\ \mathtt{F}(\mathtt{G}(\mathtt{Y}))$.

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

$(\mathcal{A}_0\to\mathcal{A}_1)\cong(\mathcal{B}_0\to\mathcal{B}_1)$.

Dependent Types

Definition. Let $\mathcal{T}$ be ***the class of all types*** on the powerset space $\mathcal{P}(\mathbb{N})$. For $\mathcal{A} \in \mathcal{T}$, an ***$\mathcal{A}$-indexed family*** of types is a function $\mathcal{B} : \mathcal{P}(\mathbb{N}) \to \mathcal{T}$, such that

$$\forall X_0, X_1.\ X_0\ \mathcal{A}\ X_1 \text{ implies } \mathcal{B}(X_0) = \mathcal{B}(X_1).$$

Definition. The ***dependent product*** of an $\mathcal{A}$-indexed family of types, $\mathcal{B}$, is defined as that relation such that

$$F_0(\prod X{:}\mathcal{A}.\mathcal{B}(X))F_1 \text{ iff}$$

$$\forall X_0, X_1.\ X_0\ \mathcal{A}\ X_1 \text{ implies } F_0(X_0)\ \mathcal{B}(X_0)\ F_1(X_1).$$

Definition. The ***dependent sum*** of an $\mathcal{A}$-indexed family of types, $\mathcal{B}$, is defined as that relation such that

$$Z_0(\sum X{:}\mathcal{A}.\mathcal{B}(X))Z_1 \text{ iff}$$

$$\exists X_0, Y_0, X_1, Y_1 [X_0 \mathcal{A} X_1\ \&\ Y_0 \mathcal{B}(X_0) Y_1\ \&$$
$$Z_0 = (X_0, Y_0)\ \&\ Z_1 = (X_1, Y_1)]$$

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*** $\mathtt{E}:\mathcal{P}$.

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 $(\mathcal{P} \to \mathcal{Q})$ means asserting an implication,
asserting $(\prod \mathtt{X}:\mathcal{A}.\mathcal{P}(\mathtt{X}))$ means asserting a universal quantification, and
asserting $(\sum \mathtt{X}:\mathcal{A}.\mathcal{B}(\mathtt{X}))$ 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
$\mathtt{Z}(\mathtt{X} \equiv_{\mathcal{A}} \mathtt{Y})\mathtt{W}$ iff $\mathtt{Z}\ \mathcal{A}\ \mathtt{X}\ \mathcal{A}\ \mathtt{Y}\ \mathcal{A}\ \mathtt{W}$.

Example: Given $\mathtt{F}:(\mathcal{A} \to (\mathcal{A} \to \mathcal{A}))$, then asserting

$$\prod \mathtt{X}:\mathcal{A}.\prod \mathtt{Y}:\mathcal{A}.\prod \mathtt{Z}:\mathcal{A}.\ \mathtt{F}(\mathtt{X})(\mathtt{F}(\mathtt{Y})(\mathtt{Z})) \equiv_{\mathcal{A}} \mathtt{F}(\mathtt{F}(\mathtt{X})(\mathtt{Y}))(\mathtt{Z})$$

means asserting that $\mathtt{F}$ is an associative operation.

Some Background References

There are many approaches to modeling λ-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.

• F. Cardone and J.R. Hindley. Lambda-Calculus and Combinators in the 20th Century. In: Volume 5, pp. 723-818, of Handbook of the History of Logic, Dov M. Gabbay and John Woods eds., North-Holland/Elsevier Science, 2009.

• Gordon D. Plotkin. Set-theoretical and other elementary models of the λ-calculus. Theoretical Computer Science, vol. 121 (1993), pp. 351-409.

• Dana S. Scott. Data types as lattices. SIAM Journal on Computing, vol. 5 (1976), pp. 522-587.

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 λ-calculus, however.

• John Myhill and John C. Shepherdson, Effective operations on partial recursive functions, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, vol. 1 (1955), pp. 310-317.

• Richard M. Friedberg and Hartley Rogers jr., Reducibility and Completeness for Sets of Integers. Mathematical Logic Quarterly, vol. 5 (1959), pp. 117-125. Some of the results of this paper are presented in abstract, Journal of Symbolic Logic, vol. 22 (1957), p. 107.

• Hartley Rogers, Jr., Theory of Recursive Functions and Effective Computability, McGraw-Hill, 1967, xix + 482 pp.

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]

- K. Bruce, A. A. Meyer, and J. C. Mitchell. The semantics of second-order lambda calculus. In G. Huet, editor. Logical Foundations of Functional Programming, pp. 273–284. Addison-Wesley, 1990.

- G. Kreisel. Interpretation of analysis by means of constructive functionals of finite type. In A. Heyting, editor, Constructivity in Mathematics, pp. 101–128. North-Holland Co., Amsterdam, 1959.

- Carl A. Gunter, and J. C. Mitchell, eds. Theoretical aspects of object-oriented programming: types, semantics, and language design. MIT Press, 1994.

- J. C. Mitchell, John C. Foundations for programming languages. MIT press, 1996, xix + 846 pp.

- M. Abadi, and G. Plotkin. A per model of polymorphism and recursive types. IEEE LICS '90, Proceedings, 1990, pp. 355–365.