

As an application, terms F and G can be constructed such that for all terms X and Y

$$\begin{aligned} FX &= XF, \\ GXY &= YG(YXG). \end{aligned}$$

2.2 Lambda definability

In the lambda calculus one can define numerals and represent numeric functions on them.

Definition 2.2.1.

1. $F^n(M)$ with $n \in \mathbb{N}$ (the set of natural numbers) and $F, M \in \Lambda$, is defined inductively as follows:

$$\begin{aligned} F^0(M) &\equiv M; \\ F^{n+1}(M) &\equiv F(F^n(M)). \end{aligned}$$

2. The *Church numerals* $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \dots$ are defined by

$$\mathbf{c}_n \equiv \lambda f x. f^n(x).$$

Proposition 2.2.2 (J. B. Rosser). Define

$$\begin{aligned} \mathbf{A}_+ &\equiv \lambda x y p q. x p (y p q); \\ \mathbf{A}_* &\equiv \lambda x y z. x (y z); \\ \mathbf{A}_{exp} &\equiv \lambda x y. y x. \end{aligned}$$

Then one has for all $n, m \in \mathbb{N}$

1. $\mathbf{A}_+ \mathbf{c}_n \mathbf{c}_m = \mathbf{c}_{n+m}$.
2. $\mathbf{A}_* \mathbf{c}_n \mathbf{c}_m = \mathbf{c}_{n \cdot m}$.
3. $\mathbf{A}_{exp} \mathbf{c}_n \mathbf{c}_m = \mathbf{c}_{(n^m)}$, except for $m = 0$ (Rosser starts at 1).

Proof. We need the following lemma.

Lemma 2.2.3.

1. $(\mathbf{c}_n x)^m(y) = x^{n*m}(y)$;
2. $(\mathbf{c}_n)^m(x) = \mathbf{c}_{(n^m)}(x)$, for $m > 0$.

Proof. 1. By induction on m . If $m = 0$, then LHS = y = RHS. Assume (1) is correct for m (Induction Hypothesis: *IH*). Then

$$\begin{aligned}
(\mathbf{c}_n x)^{m+1}(y) &= \mathbf{c}_n x((\mathbf{c}_n x)^m(y)) \\
&\stackrel{IH}{=} \mathbf{c}_n x(x^{n*m}(y)) \\
&= x^n(x^{n*m}(y)) \\
&\equiv x^{n+n*m}(y) \\
&\equiv x^{n*(m+1)}(y).
\end{aligned}$$

2. By induction on $m > 0$. If $m = 1$, then LHS $\equiv \mathbf{c}_n x \equiv$ RHS. If (2) is correct for m , then

$$\begin{aligned}
\mathbf{c}_n^{m+1}(x) &= \mathbf{c}_n(\mathbf{c}_n^m(x)) \\
&\stackrel{IH}{=} \mathbf{c}_n(\mathbf{c}_{(n^m)}(x)) \\
&= \lambda y. (\mathbf{c}_{(n^m)}(x))^n(y) \\
&\stackrel{(1)}{=} \lambda y. x^{n*n^m}(y) \\
&= \mathbf{c}_{(n^{m+1})}x. \blacksquare
\end{aligned}$$

■

Now the proof of the proposition.

1. By induction on m .
2. Use the lemma (1).
3. By the lemma (2) we have for $m > 0$

$$\mathbf{A}_{exp} \mathbf{c}_n \mathbf{c}_m = \mathbf{c}_m \mathbf{c}_n = \lambda x. \mathbf{c}_n^m(x) = \lambda x. \mathbf{c}_{(n^m)}x = \mathbf{c}_{(n^m)},$$

since $\lambda x. Mx = M$ if $M = \lambda y. M'[y]$ and $x \notin FV(M)$. Indeed,

$$\begin{aligned}
\lambda x. Mx &= \lambda x. (\lambda y. M'[y])x \\
&= \lambda x. M'[x] \\
&\equiv \lambda y. M'[y] \\
&= M. \blacksquare
\end{aligned}$$

■

We have seen that the functions plus, times and exponentiation on \mathbb{N} can be represented in the λ -calculus using Church's numerals. We will show that all computable (recursive) functions can be represented.

Boolean truth values and a conditional can be represented in the λ -calculus.

Definition 2.2.4 (Booleans, conditional).

1. $\mathbf{true} \equiv \lambda xy.x$, $\mathbf{false} \equiv \lambda xy.y$.
2. If B is a Boolean, i.e. a term that is either \mathbf{true} , or \mathbf{false} , then

if B then P else Q

can be represented by BPQ . Indeed, $\mathbf{true}PQ = P$ and $\mathbf{false}PQ = Q$.

Definition 2.2.5 (Pairing). For $M, N \in \Lambda$ write

$$[M, N] \equiv \lambda z.zMN.$$

Then

$$[M, N] \mathbf{true} = M$$

$$[M, N] \mathbf{false} = N$$

and hence $[M, N]$ can serve as an ordered pair.

Definition 2.2.6.

1. A *numeric function* is a map $f : \mathbb{N}^p \rightarrow \mathbb{N}$ for some p .
2. A numeric function f with p arguments is called *λ -definable* if one has for some combinator F

$$F \mathbf{c}_{n_1} \dots \mathbf{c}_{n_p} = \mathbf{c}_{f(n_1, \dots, n_p)} \quad (1)$$

for all $n_1, \dots, n_p \in \mathbb{N}$. If (1) holds, then f is said to be *λ -defined* by F .

Definition 2.2.7.

1. The *initial functions* are the numeric functions U_r^i, S^+, Z defined by:

$$\begin{aligned} U_r^i(x_1, \dots, x_r) &= x_i, \quad 1 \leq i \leq r; \\ S^+(n) &= n + 1; \\ Z(n) &= 0. \end{aligned}$$

2. Let $P(n)$ be a numeric relation. As usual

$$\mu m.P(m)$$

denotes the least number m such that $P(m)$ holds if there is such a number; otherwise it is undefined.

As we know from Chapter 2 in this handbook, the class \mathcal{R} of recursive functions is the smallest class of numeric functions that contains all

initial functions and is closed under composition, primitive recursion and minimalization. So \mathcal{R} is an inductively defined class. The proof that all recursive functions are λ -definable is by a corresponding induction argument. The result is originally due to Kleene (1936).

Lemma 2.2.8. *The initial functions are λ -definable.*

Proof. Take as defining terms

$$\begin{aligned} \mathbf{U}_p^i &\equiv \lambda x_1 \cdots x_p. x_i; \\ \mathbf{S}^+ &\equiv \lambda xyz. y(xyz) \quad (= \mathbf{A}_+ \mathbf{c}_1); \\ \mathbf{Z} &\equiv \lambda x. \mathbf{c}_0. \blacksquare \end{aligned}$$

■

Lemma 2.2.9. *The λ -definable functions are closed under composition.*

Proof. Let g, h_1, \dots, h_m be λ -defined by G, H_1, \dots, H_m respectively. Then

$$f(\vec{n}) = g(h_1(\vec{n}), \dots, h_m(\vec{n}))$$

is λ -defined by

$$F \equiv \lambda \vec{x}. G(H_1 \vec{x}) \dots (H_m \vec{x}). \blacksquare$$

■

Lemma 2.2.10. *The λ -definable functions are closed under primitive recursion.*

Proof. Let f be defined by

$$\begin{aligned} f(0, \vec{n}) &= g(\vec{n}) \\ f(k+1, \vec{n}) &= h(f(k, \vec{n}), k, \vec{n}) \end{aligned}$$

where g, h are λ -defined by G, H respectively. We have to show that f is λ -definable. For notational simplicity we assume that there are no parameters \vec{n} (hence $G = \mathbf{c}_{f(0)}$.) The proof for general \vec{n} is similar.

If k is not an argument of h , then we have the scheme of iteration. Iteration can be represented easily in the λ -calculus, because the Church numerals are iterators. The construction of the representation of f is done

in two steps. First primitive recursion is reduced to iteration using ordered pairs; then iteration is represented. Here are the details. Consider

$$T \equiv \lambda p.[S^+(p\mathbf{true}), H(p\mathbf{false})(p\mathbf{true})].$$

Then for all k one has

$$\begin{aligned} T([\mathbf{c}_k, \mathbf{c}_{f(k)}]) &= [\mathbf{f}S^+ \mathbf{c}_k, H\mathbf{c}_{f(k)}\mathbf{c}_k] \\ &= [\mathbf{c}_{k+1}, \mathbf{c}_{f(k+1)}]. \end{aligned}$$

By induction on k it follows that

$$[\mathbf{c}_k, \mathbf{c}_{f(k)}] = T^k[\mathbf{c}_0, \mathbf{c}_{f(0)}].$$

Therefore

$$\mathbf{c}_{f(k)} = \mathbf{c}_k T[\mathbf{c}_0, \mathbf{c}_{f(0)}] \mathbf{false},$$

and f can be λ -defined by

$$F \equiv \lambda k.kT[\mathbf{c}_0, G] \mathbf{false}. \blacksquare$$

Lemma 2.2.11. *The λ -definable functions are closed under minimalization.*

Proof. Let f be defined by $f(\vec{n}) = \mu m[g(\vec{n}, m) = 0]$, where $\vec{n} = n_1, \dots, n_k$ and g is λ -defined by G . We have to show that f is λ -definable. Define

$$\mathbf{zero} \equiv \lambda n.n(\mathbf{true} \mathbf{false})\mathbf{true}.$$

Then

$$\begin{aligned} \mathbf{zero} \mathbf{c}_0 &= \mathbf{true}, \\ \mathbf{zero} \mathbf{c}_{n+1} &= \mathbf{false}. \end{aligned}$$

By Corollary 2.1.10 there is a term H such that

$$H\vec{n}y = \mathbf{if} (\mathbf{zero}(G\vec{n}y)) \mathbf{then} y \mathbf{else} H\vec{n}(S^+y).$$

Set $F = \lambda \vec{n}.H\vec{x}\mathbf{c}_0$. Then F λ -defines f :

$$\begin{aligned} F\mathbf{c}_{\vec{n}} &= H\mathbf{c}_{\vec{n}}\mathbf{c}_0 \\ &= \mathbf{c}_0, && \text{if } G\mathbf{c}_{\vec{n}}\mathbf{c}_0 = \mathbf{c}_0, \\ &= H\mathbf{c}_{\vec{n}}\mathbf{c}_1 && \text{else;} \\ &= \mathbf{c}_1, && \text{if } G\mathbf{c}_{\vec{n}}\mathbf{c}_1 = \mathbf{c}_0, \\ &= H\mathbf{c}_{\vec{n}}\mathbf{c}_2 && \text{else;} \\ &= \mathbf{c}_2, && \text{if } \dots \\ &= \dots \end{aligned}$$

Here $\mathbf{c}_{\vec{n}}$ stands for $\mathbf{c}_{n_1} \dots \mathbf{c}_{n_k}$. \blacksquare

Theorem 2.2.12. *All recursive functions are λ -definable.*

Proof. By 2.2.8-2.2.11. ■

The converse also holds. The idea is that if a function is λ -definable, then its graph is recursively enumerable because equations derivable in the λ -calculus can be enumerated. It then follows that the function is recursive. So for numeric functions we have f is recursive iff f is λ -definable. Moreover also for partial functions a notion of λ -definability exists and one has ψ is partial recursive iff ψ is λ -definable. The notions λ -definable and recursive both are intended to be formalizations of the intuitive concept of computability. Another formalization was proposed by Turing in the form of Turing computable. The equivalence of the notions recursive, λ -definable and Turing computable (for the latter see besides the original Turing, 1937, e.g., Davis 1958) Davis provides some evidence for the Church–Turing thesis that states that ‘recursive’ is the proper formalization of the intuitive notion ‘computable’.

We end this subsection with some undecidability results. First we need the coding of λ -terms. Remember that the collection of variables is $\{v, v', v'', \dots\}$.

Definition 2.2.13.

1. Notation. $v^{(0)} = v$; $v^{(n+1)} = v^{(n)'$.
2. Let $\langle \cdot, \cdot \rangle$ be a recursive coding of pairs of natural numbers as a natural number. Define

$$\begin{aligned} \sharp(v^{(n)}) &= \langle 0, n \rangle; \\ \sharp(MN) &= \langle 2, \langle \sharp(M), \sharp(N) \rangle \rangle; \\ \sharp(\lambda x.M) &= \langle 3, \langle \sharp(x), \sharp(M) \rangle \rangle. \end{aligned}$$

3. Notation

$$\ulcorner M \urcorner = \mathbf{c}_{\sharp M}.$$

Definition 2.2.14. Let $\mathcal{A} \subseteq \Lambda$.

1. \mathcal{A} is closed under $=$ if

$$M \in \mathcal{A}, \lambda \vdash M = N \Rightarrow N \in \mathcal{A}.$$

2. \mathcal{A} is non-trivial if $\mathcal{A} \neq \emptyset$ and $\mathcal{A} \neq \Lambda$.
3. \mathcal{A} is recursive if $\sharp\mathcal{A} = \{\sharp M \mid M \in \mathcal{A}\}$ is recursive.

The following result due to Scott is quite useful for proving undecidability results.

Theorem 2.2.15. *Let $\mathcal{A} \subseteq \Lambda$ be non-trivial and closed under $=$. Then \mathcal{A} is not recursive.*

Proof. (J. Terlouw) Define

$$\mathcal{B} = \{M \mid M^\top M^\top \in \mathcal{A}\}.$$

Suppose \mathcal{A} is recursive; then by the effectiveness of the coding also \mathcal{B} is recursive (indeed, $n \in \mathcal{B} \Leftrightarrow \langle 2, \langle n, \mathbf{c}_n \rangle \rangle \in \mathcal{A}$). It follows that there is an $F \in \Lambda^0$ with

$$\begin{aligned} M \in \mathcal{B} &\Leftrightarrow F^\top M^\top = \mathbf{c}_0; \\ M \notin \mathcal{B} &\Leftrightarrow F^\top M^\top = \mathbf{c}_1. \end{aligned}$$

Let $M_0 \in \mathcal{A}, M_1 \notin \mathcal{A}$. We can find a $G \in \Lambda$ such that

$$\begin{aligned} M \in \mathcal{B} &\Leftrightarrow G^\top M^\top = M_1 \notin \mathcal{A}, \\ M \notin \mathcal{B} &\Leftrightarrow G^\top M^\top = M_0 \in \mathcal{A}. \end{aligned}$$

[Take $Gx = \mathbf{if\ zero}(Fx) \mathbf{then} M_1 \mathbf{else} M_0$, with \mathbf{zero} defined in the proof of 2.2.11.] In particular

$$\begin{aligned} G \in \mathcal{B} &\Leftrightarrow G^\top G^\top \notin \mathcal{A} \Leftrightarrow_{\text{Def}} G \notin \mathcal{B}, \\ G \notin \mathcal{B} &\Leftrightarrow G^\top G^\top \in \mathcal{A} \Leftrightarrow_{\text{Def}} G \in \mathcal{B}, \end{aligned}$$

a contradiction. ■

The following application shows that the lambda calculus is not a decidable theory.

Corollary 2.2.16 (Church). *The set*

$$\{M \mid M = \mathbf{true}\}$$

is not recursive.

Proof. Note that the set is closed under $=$ and is nontrivial. ■

2.3 Reduction

There is a certain asymmetry in the basic scheme (β). The statement

$$(\lambda x.x^2 + 1)3 = 10$$

can be interpreted as ‘10 is the result of computing $(\lambda x.x^2 + 1)3$ ’, but not vice versa. This computational aspect will be expressed by writing

$$(\lambda x.x^2 + 1)3 \rightarrow 10$$

which reads ‘ $(\lambda x.x^2 + 1)3$ reduces to 10’.