$$
\begin{aligned}
&= \lambda z z'.z z' \\
&\equiv \lambda y z.y z \\
&\equiv 1.
\end{aligned}
$$

3. For implementations of the λ-calculus the machine has to deal with this so called α-conversion. A good way of doing this is provided by the 'name-free notation' of N.G. de Bruijn, see Barendregt (1984), Appendix C. In this notation $\lambda x(\lambda y.xy)$ is denoted by $\lambda(\lambda 21)$, the 2 denoting a variable bound 'two lambdas above'.

The following result provides one way to represent recursion in the λ-calculus.

Theorem 2.1.9 (Fixed point theorem).

1. $\forall F \exists X FX = X$.
(This means that for all $F \in \Lambda$ *there is an* $X \in \Lambda$ *such that* $\lambda \vdash FX = X$.*)*

2. There is a fixed point combinator

$$\mathsf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

such that

$$\forall F\ F(\mathsf{Y}F) = \mathsf{Y}F.$$

Proof. 1. Define $W \equiv \lambda x.F(xx)$ and $X \equiv WW$. Then
$X \equiv WW \equiv (\lambda x.F(xx))W = F(WW) \equiv FX$.

2. By the proof of (1). Note that
$\mathsf{Y}F = (\lambda x.F(xx))(\lambda x.F(xx)) \equiv X$. ■

■

Corollary 2.1.10. *Given a term* $C \equiv C[f,x]$ *possibly containing the displayed free variables, then*

$$\exists F \forall X\ FX = C[F,X].$$

Here $C[F,X]$ *is of course the substitution result* $C[f := F][x := X]$.

Proof. Indeed, we can construct F by supposing it has the required property and calculating back:

$$
\begin{array}{rrcl}
 & \forall X\ FX & = & C[F,X] \\
\Leftarrow & Fx & = & C[F,x] \\
\Leftarrow & F & = & \lambda x.C[F,x] \\
\Leftarrow & F & = & (\lambda f x.C[f,x])F \\
\Leftarrow & F & \equiv & \mathsf{Y}(\lambda f x.C[f,x]). \blacksquare
\end{array}
$$

■

This also holds for more arguments: $\exists F \forall \vec{x}\ F\vec{x} = C[F,\vec{x}]$.

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

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

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:

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

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

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

Proposition 2.2.2 (J. B. Rosser). *Define*

$$\begin{array}{rcl} \mathsf{A}_+ & \equiv & \lambda xypq.xp(ypq); \\ \mathsf{A}_* & \equiv & \lambda xyz.x(yz); \\ \mathsf{A}_{exp} & \equiv & \lambda xy.yx. \end{array}$$

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

1. $\mathsf{A}_+\mathbf{c}_n\mathbf{c}_m = \mathbf{c}_{n+m}$.

2. $\mathsf{A}_*\mathbf{c}_n\mathbf{c}_m = \mathbf{c}_{n.m}$.

3. $\mathsf{A}_{exp}\mathbf{c}_n\mathbf{c}_m = \mathbf{c}_{(n^m)}$, *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

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

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

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

∎

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,

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

∎

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 **true**, or **false**, then

   $$\mathbf{if}\ B\ \mathbf{then}\ P\ \mathbf{else}\ Q$$

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

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

   for all $n_1, \ldots, n_p \in \mathbb{N}$. 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:

   $$\begin{array}{rcl} U_r^i(x_1, \ldots, x_r) & = & x_i, \quad 1 \le i \le r; \\ S^+(n) & = & n + 1; \\ Z(n) & = & 0. \end{array}$$

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

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

∎

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

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

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

is λ-defined by

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

∎

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

Proof. Let f be defined by

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

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

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

Then for all k one has

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

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

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

$\blacksquare$

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 $\vec{n} = n_1, \ldots, n_k$ 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

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

Set $F = \lambda\vec{n}.H\vec{x}\mathbf{c}0$. Then F λ-defines f:

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

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

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 $\{v, v', v'', \ldots\}$.

Definition 2.2.13.

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

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

3. Notation

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

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

1. *A is closed under* = if

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

2. *A is non-trivial* if $\mathcal{A} \neq \emptyset$ and $\mathcal{A} \neq \Lambda$.
3. *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.