

variables x_1, x_2, \dots . Rather than referring to specific variables, such as x_{17} , we shall use letters x, y, z, f, g, h, \dots as arbitrary variables. We now define *terms* of the λ -calculus:

1. each variable is a term;
2. if ϕ and ψ are terms, so is $(\phi\psi)$;
3. if ϕ is a term and x is a variable, then $(\lambda_x\phi)$ is a term;
4. all terms are built up according to the above rules.

We have inserted parentheses to avoid ambiguity. However, when there is no danger of ambiguity we may omit them.

We define a *free* occurrence of a variable in a term as follows:

1. a variable x occurs freely in the term x ;
2. if x occurs freely in ϕ or ψ , then it occurs freely in $\phi\psi$;
3. if y is a variable distinct from x and y occurs freely in ϕ , then it also occurs freely in $\lambda_x\phi$; however, x does *not* occur freely in $\lambda_x\phi$.

If the variable x occurs freely in the term ϕ , then it said to be *bound* to λ_x in $\lambda_x\phi$. We often write the term ϕ as $\phi(x)$ to indicate the possible free occurrence of x in ϕ . This has the advantage that we can write $\phi(\alpha)$ for the result of substituting α for x in ϕ . But we are only *allowed to substitute* α for x in $\phi(x)$ if no variable occurring freely in α becomes bound in $\phi(\alpha)$.

In addition to the usual properties of equality — reflexivity, symmetry, transitivity and the rule allowing substitution of equals — we postulate the following three rules:

R1. $(\lambda_x\phi(x))\alpha = \phi(\alpha)$, if we are allowed to substitute α for x in $\phi(x)$;

R2. $\lambda_x(\phi'x) = \phi$, if x is not free in ϕ (possibly because x does not occur in ϕ at all);

R3. $\lambda_x\phi(x) = \lambda_y\phi(y)$, if we are allowed to substitute y for x in $\phi(x)$.

Here are some examples to illustrate these ideas, which the reader may skip if she has already understood them.

1. $(\lambda_x(y'x))\alpha$ is a term with the first occurrence of x not free, it being bound to λ_x , and the second occurrence of x free.

2. Suppose $\phi(x)$ and $\psi(x)$ are terms not containing λ_y and suppose we know that $\phi(x) = \psi(x)$. Then $\lambda_x\phi(x)`y = \lambda_x\psi(x)`y$, by the properties of equality, hence $\phi(y) = \psi(y)$ by R1.
3. $\lambda_f(g`f) = g$ by R2 (assuming $g \neq f$) and $(\lambda_g g)`f = f$ by R1.
4. What can go wrong if we disregard the ‘if’ clauses of the rules? From 3 we see that

$$(\lambda_g(\lambda_f(g`f)))`f = (\lambda_g g)`f = f.$$

But disregarding the ‘if’ clause in R1 would yield

$$(\lambda_g(\lambda_f(g`f)))`f = \lambda_f(f`f)$$

(taking $x \equiv g$ and $\phi(x) \equiv \lambda_f(g`f)$). But f and $\lambda_f(f`f)$ are not the same, as is seen by applying both to a , say. $f`a$ depends on f , but $(\lambda_f(f`f))`a = a`a$ does not; f in the last term is a *dummy* variable.

We shall now show how to do arithmetic in the lambda calculus. First note that any two functions f and g can be *composed* to form $f \circ g$, where $(f \circ g)`x = f`g`x$, hence $f \circ g = \lambda_x(f`g`x)$ by R2. In particular, $f \circ f$ is usually written f^2 , the *iterate* of f , so $f^2 = \lambda_x(f`f`x)$. Church had the idea that the number 2 should be defined as the *process of iteration*, as the function which assigns f^2 to f , that is

$$2 = \lambda_f(f^2) = \lambda_f(\lambda_x(f`f`x)).$$

Since it is customary to think of f^0 as the identity function and of f^1 as f , we may also define

$$0 = \lambda_f(\lambda_x x), \quad 1 = \lambda_f(\lambda_x(f`x)) = \lambda_f f,$$

hence $0 = \lambda_f 1$ since $\lambda_f f = \lambda_x x$ by R3. The reader is invited to define the number 3 in the lambda calculus. In general, $n = \lambda_f(f^n)$, where $f^n = f \circ f \circ \dots \circ f$ is the n th iterate of f . More precisely, f^n is defined inductively by $f^{\Sigma`n} = f \circ f^n$, where $\Sigma`n$ is the successor of n . Thus $n`f = f^n$.

Substituting m for f in this equation, we have $n`m = m^n$. But this tells us how to define *exponentiation* in the lambda calculus. In particular, $m^0 = 0`m = 1$. Since $(m \times n)`f = f^{m \times n} = (f^n)^m = m`n`f = (m \circ n)`f$, we may define *multiplication* by $m \times n = m \circ n = \lambda_x(m`n`x)$. For *addition* we have

$$(m + n)`f = f^{m+n} = f^m \circ f^n = (m`f) \circ (n`f),$$

so that $m + n = \lambda_f((m`f) \circ (n`f))$.

We can now prove some of the basic laws of arithmetic. For example, since composition of functions is associative, multiplication of numbers is associative. On the other hand, composition of functions is not commutative, yet multiplication of numbers is; this must be proved by mathematical induction.

Of considerable interest in computer science is the following:

Theorem 22.1. (Fixpoint Theorem)

For any term ϕ , there is a term α such that $\phi^{\cdot}\alpha = \alpha$ can be proved.

This implies, in particular, that it is not possible to incorporate a term into the lambda calculus which corresponds to the negation symbol in logic.

Proof: Let $\beta = \lambda_x(\phi^{\cdot}(x^{\cdot}x))$ and $\alpha = \beta^{\cdot}\beta$. Then

$$\alpha = \beta^{\cdot}\beta = (\lambda_x(\phi^{\cdot}(x^{\cdot}x)))^{\cdot}\beta = \phi^{\cdot}(\beta^{\cdot}\beta) = \phi^{\cdot}\alpha.$$

Surprisingly, it is possible to get rid of the λ -abstraction (and all bound variables) in the lambda calculus. This discovery goes back to Schönfinkel (1924). In fact, if we write

$$I = \lambda_x x \quad (\text{identity}),$$

$$K = \lambda_x \lambda_y x \quad (\text{constancy operator}),$$

$$S = \lambda_x \lambda_y \lambda_z ((x^{\cdot}z)^{\cdot}(y^{\cdot}z)) \quad (\text{Schönfinkel operator}),$$

we have

$$I^{\cdot}x = x \quad (\text{thus } I = 1),$$

$$(K^{\cdot}x)^{\cdot}y = x \quad (\text{thus } K^{\cdot}x \text{ is the function with constant value } x),$$

$$((S^{\cdot}f)^{\cdot}g)^{\cdot}x = (f^{\cdot}x)^{\cdot}(g^{\cdot}x).$$

We define *combinators* as follows:

1. all variables are combinators;
2. I , K , and S are combinators;
3. if ϕ and ψ are combinators, so is $\phi^{\cdot}\psi$;
4. all combinators are obtained from the above rules.

Theorem 22.2. (Schönfinkel)

Every term of the lambda calculus is provably equal to a combinator.

Proof: In view of rules (1) to (3), this will follow if we show that, if ϕ is equal to a combinator, so is $\lambda_x\phi$. We prove this by induction on the length of ϕ .

Since $\lambda_x x = I$, $\lambda_x y = K^{\cdot}y$, $\lambda_x I = K^{\cdot}I$, $\lambda_x K = K^{\cdot}K$ and $\lambda_x S = K^{\cdot}S$, we know that the induction hypothesis holds for all combinators ϕ of length 1.