1 The λ-calculus

1A Introduction

What is usually called λ-calculus is a collection of several formal systems, based on a notation invented by Alonzo Church in the 1930s. They are designed to describe the most basic ways that operators or functions can be combined to form other operators.

In practice, each λ-system has a slightly different grammatical structure, depending on its intended use. Some have extra constant-symbols, and most have built-in syntactic restrictions, for example type-restrictions. But to begin with, it is best to avoid these complications; hence the system presented in this chapter will be the 'pure' one, which is syntactically the simplest.

To motivate the λ-notation, consider the everyday mathematical expression '$x - y$'. This can be thought of as defining either a function f of x or a function g of y;

$$f(x) = x - y, \qquad\qquad g(y) = x - y,$$

or

$$f: \; x \mapsto x - y, \qquad\qquad g: \; y \mapsto x - y.$$

And there is a need for a notation that gives f and g different names in some systematic way. In practice, mathematicians usually avoid this need by various 'ad-hoc' special notations, but these can get very clumsy when higher-order functions are involved (functions which act on other functions).

Church's notation is a systematic way of constructing, for each expression involving 'x', a notation for the corresponding function of x (and similarly for 'y', etc.). Church introduced 'λ' as an auxiliary symbol and wrote

$$f \;=\; \lambda x\,.\,x-y \qquad\qquad g \;=\; \lambda y\,.\,x-y.$$

For example, consider the equations

$$f(0) \;=\; 0-y, \qquad\qquad f(1) \;=\; 1-y.$$

In the λ-notation these become

$$(\lambda x\,.\,x-y)(0) \;=\; 0-y, \qquad\qquad (\lambda x\,.\,x-y)(1) \;=\; 1-y.$$

These equations are clumsier than the originals, but do not be put off by this; the λ-notation is principally intended for denoting higher-order functions, not just functions of numbers, and for this it turns out to be no worse than others.[1] The main point is that this notation is systematic, and therefore more suitable for incorporation into a programming language.

The λ-notation can be extended to functions of more than one variable. For example, the expression '$x-y$' determines two functions h and k of two variables, defined by

$$h(x,y) \;=\; x-y, \qquad\qquad k(y,x) \;=\; x-y.$$

These can be denoted by

$$h \;=\; \lambda xy\,.\,x-y, \qquad\qquad k \;=\; \lambda yx\,.\,x-y.$$

However, we can avoid the need for a special notation for functions of several variables by using functions whose values are not numbers but other functions. For example, instead of the two-place function h above, consider the one-place function $h^\star$ defined by

$$h^\star \;=\; \lambda x\,.\,(\lambda y\,.\,x-y).$$

For each number a, we have

$$h^\star(a) \;=\; \lambda y\,.\,a-y;$$

hence for each pair of numbers a, b,

$$\begin{aligned}(h^\star(a))(b) &= (\lambda y\,.\,a-y)(b)\\ &= a-b\\ &= h(a,b).\end{aligned}$$

[1] For example, one fairly common notation in mathematics is $f = x \mapsto x-y$, which is essentially just the λ-notation in disguise, with '$x \mapsto$' instead of 'λx'.

Thus $h^\star$ can be viewed as 'representing' h. For this reason, we shall largely ignore functions of more than one variable in this book.

From now on, '*function*' will mean '*function of one variable*' unless explicitly stated otherwise. (The use of $h^\star$ instead of h is usually called *currying*.[2])

Having looked at λ-notation in an informal context, let us now construct a formal system of λ-calculus.

Definition 1.1 (λ-terms) Assume that there is given an infinite sequence of expressions $\mathbf{v}_0$, $\mathbf{v}_{00}$, $\mathbf{v}_{000}$, ... called *variables*, and a finite, infinite or empty sequence of expressions called *atomic constants*, different from the variables. (When the sequence of atomic constants is empty, the system will be called *pure*, otherwise *applied*.) The set of expressions called λ-*terms* is defined inductively as follows:

(a) all variables and atomic constants are λ-terms (called *atoms*);
(b) if M and N are any λ-terms, then (MN) is a λ-term (called an *application*);
(c) if M is any λ-term and x is any variable, then $(\lambda x . M)$ is a λ-term (called an *abstraction*).

Example 1.2 (Some λ-terms)

(a) $(\lambda \mathbf{v}_0 . (\mathbf{v}_0 \mathbf{v}_{00}))$ is a λ-term.

If x, y, z are any distinct variables, the following are λ-terms:

(b) $(\lambda x . (xy))$, (d) $(x\,(\lambda x . (\lambda x . x)))$,

(c) $((\lambda y . y)(\lambda x . (xy)))$, (e) $(\lambda x . (yz))$.

In (d), there are two occurrences of λx in one term; this is allowed by Definition 1.1, though not encouraged in practice. Part (e) shows a term of form $(\lambda x . M)$ such that x does not occur in M; this is called a *vacuous abstraction*, and such terms denote constant functions (functions whose output is the same for all inputs).

By the way, the expression 'λ' by itself is not a term, though it may occur in terms; similarly the expression 'λx' is not a term.

[2] Named after Haskell Curry, one of the inventors of combinatory logic. Curry always insisted that he got the idea of using $h^\star$ from M. Schönfinkel's [Sch24] (see [CF58, pp. 8, 10]), but most workers seem to prefer to pronounce 'currying' rather than 'schönfinkeling'. The idea also appeared in 1893 in [Fre93, Vol. 1, Section 4].

Notation 1.3 *Capital letters* will denote arbitrary λ-terms in this chapter. Letters 'x', 'y', 'z', 'u', 'v', 'w' will denote variables throughout the book, and distinct letters will denote distinct variables unless stated otherwise.

Parentheses will be omitted in such a way that, for example, '$MNPQ$' will denote the term $(((MN)P)Q)$. (This convention is called *association to the left*.) Other abbreviations will be

$$\begin{array}{lll} \lambda x.PQ & \text{for} & (\lambda x.(PQ)), \\ \lambda x_1x_2\ldots x_n.M & \text{for} & (\lambda x_1.(\lambda x_2.(\ldots(\lambda x_n.M)\ldots))). \end{array}$$

Syntactic identity of terms will be denoted by '$\equiv$'; in other words

$$M \equiv N$$

will mean that M is exactly the same term as N. (The symbol '$=$' will be used in formal theories of equality, and for identity of objects that are not terms, such as numbers.) It will be assumed of course that if $MN \equiv PQ$ then $M \equiv P$ and $N \equiv Q$, and if $\lambda x.M \equiv \lambda y.P$ then $x \equiv y$ and $M \equiv P$. It will also be assumed that variables are distinct from constants, and applications are distinct from abstractions, etc. Such assumptions are always made when languages are defined, and will be left unstated in future.

The cases $k = 0$, $n = 0$ in statements like '$P \equiv MN_1\ldots N_k$ $(k \geq 0)$' or 'T has form $\lambda x_1\ldots x_n.PQ$ $(n \geq 0)$' will mean '$P \equiv M$' or 'T has form PQ'.

'λ' will often be used carelessly to mean 'λ-calculus in general'.

'*Iff*' will be used for 'if and only if'.

Exercise 1.4 * Insert the full number of parentheses and λ's into the following abbreviated λ-terms:

(a)	$xyz(yx)$,	(d)	$(\lambda u.vuu)zy$,
(b)	$\lambda x.uxy$,	(e)	$ux(yz)(\lambda v.vy)$,
(c)	$\lambda u.u(\lambda x.y)$,	(f)	$(\lambda xyz.xz(yz))uvw$.

Informal interpretation 1.5 Not all systems based on λ-calculus use all the terms allowed by Definition 1.1, and in most systems, some terms are left uninterpreted, as we shall see later. But the interpretations of those λ-terms which are interpreted may be given as follows, roughly speaking.

In general, if M has been interpreted as a function or operator, then (MN) is interpreted as the result of applying M to argument N, provided this result exists.[3]

A term $(\lambda x.M)$ represents the operator or function whose value at an argument N is calculated by substituting N for x in M.

For example, $\lambda x.x(xy)$ represents the operation of applying a function twice to an object denoted by y; and the equation

$$(\lambda x.x(xy))N \;=\; N(Ny)$$

holds for all terms N, in the sense that both sides have the same interpretation.

For a second example, $\lambda x.y$ represents the constant function that takes the value y for all arguments, and the equation

$$(\lambda x.y)N \;=\; y$$

holds in the same sense as before.

This is enough on interpretation for the moment; but more will be said in Chapter 3, Discussion 3.27.

1B Term-structure and substitution

The main topic of the chapter will be a formal procedure for calculating with terms, that will closely follow their informal meaning. But before defining it, we shall need to know how to substitute terms for variables, and this is not entirely straightforward. The present section covers the technicalities involved. The details are rather boring, and the reader who is just interested in main themes should read only up to Definition 1.12 and then go to the next section.

By the way, in Chapter 2 a simpler system called combinatory logic will be described, which will avoid most of the boring technicalities; but for this gain there will be a price to pay.

Definition 1.6 The *length* of a term M (called $\mathit{lgh}(M)$) is the total number of occurrences of atoms in M. In more detail, define

[3] The more usual notation for function-application is $M(N)$, but historically (MN) has become standard in λ-calculus. This book uses the (MN) notation for formal terms (following Definition 1.1(b)), but reverts to the common notation, e.g. $f(a)$, in informal discussions of functions of numbers, etc.

(a)	$lgh(a)$	$=$	1	for atoms a;
(b)	$lgh(MN)$	$=$	$lgh(M) + lgh(N)$;	
(c)	$lgh(\lambda x . M)$	$=$	$1 + lgh(M)$.	

The phrase '*induction on* M' will mean 'induction on $lgh(M)$'.

For example, if $M \equiv x(\lambda y . yux)$ then $lgh(M) = 5$.

Definition 1.7 For λ-terms P and Q, the relation P *occurs in* Q (or P *is a subterm of* Q, or Q *contains* P) is defined by induction on Q, thus:

(a) P occurs in P;
(b) if P occurs in M or in N, then P occurs in (MN);
(c) if P occurs in M or $P \equiv x$, then P occurs in $(\lambda x . M)$.

The meaning of '*an occurrence of* P *in* Q' is assumed to be intuitively clear. For example, in the term $((xy)(\lambda x . (xy)))$ there are two occurrences of (xy) and three occurrences of x.[4]

Exercise 1.8 * (Hint: in each part below, first write the given terms in full, showing all parentheses and λ's.)

(a) Mark all the occurrences of xy in the term $\lambda xy . xy$.
(b) Mark all the occurrences of uv in $x(uv)(\lambda u . v(uv))uv$.
(c) Does $\lambda u . u$ occur in $\lambda u . uv$?

Definition 1.9 (Scope) For a particular occurrence of $\lambda x . M$ in a term P, the occurrence of M is called the *scope* of the occurrence of λx on the left.

Example 1.10 Let

$$P \equiv (\lambda y . yx(\lambda x . y(\lambda y . z)x))vw.$$

The scope of the leftmost λy in P is $yx(\lambda x . y(\lambda y . z)x)$, the scope of λx is $y(\lambda y . z)x$, and that of the rightmost λy is z.

Definition 1.11 (Free and bound variables) An occurrence of a variable x in a term P is called

- *bound* if it is in the scope of a λx in P,

[4] The reader who wants more precision can define an occurrence of P in Q to be a pair $\langle P, p\rangle$ where p is some indicator of the position at which P occurs in Q. There are several definitions of suitable position indicators in the literature, for example in [Ros73, p. 167] or [Hin97, pp. 140–141]. But it is best to avoid such details for as long as possible.

- *bound and binding*, iff it is the x in λx,
- *free* otherwise.

If x has at least one binding occurrence in P, we call x a *bound variable of* P. If x has at least one free occurrence in P, we call x a *free variable of* P. The set of all free variables of P is called

$$\mathrm{FV}(P).$$

A *closed term* is a term without any free variables.

Examples Consider the term $xv(\lambda yz.yv)w$: this is really

$$\Big(\big((xv)(\lambda y.(\lambda z.(yv)))\big)\, w\Big),$$

and in it the x on the left is free, the leftmost v is free, the leftmost y is both bound and binding, the only z is the same, the rightmost y is bound but not binding, the rightmost v is free, and the only w is free.

In the term P in Example 1.10, all four y's are bound, the leftmost and rightmost y's are also binding, the left-hand x is free, the central x is bound and binding, the right-hand x is bound but not binding, and z, v, w are free; hence

$$\mathrm{FV}(P) = \{x, z, v, w\}.$$

Note that x is both a free and a bound variable of P; this is not normally advisable in practice, but is allowed in order to keep the definition of 'λ-term' simple.

Definition 1.12 (Substitution) For any M, N, x, define $[N/x]M$ to be the result of substituting N for every free occurrence of x in M, and changing bound variables to avoid clashes. The precise definition is by induction on M, as follows (after [CF58, p. 94]).

$$
\begin{array}{lll}
\text{(a)}\ [N/x]\,x & \equiv N; & \\
\text{(b)}\ [N/x]\,a & \equiv a & \text{for all atoms } a \not\equiv x; \\
\text{(c)}\ [N/x](PQ) & \equiv \big([N/x]P\ [N/x]Q\big); & \\
\text{(d)}\ [N/x](\lambda x.P) & \equiv \lambda x.P; & \\
\text{(e)}\ [N/x](\lambda y.P) & \equiv \lambda y.P & \text{if } x \notin \mathrm{FV}(P); \\
\text{(f)}\ [N/x](\lambda y.P) & \equiv \lambda y.[N/x]P & \text{if } x \in \mathrm{FV}(P) \text{ and } y \notin \mathrm{FV}(N); \\
\text{(g)}\ [N/x](\lambda y.P) & \equiv \lambda z.[N/x][z/y]P & \text{if } x \in \mathrm{FV}(P) \text{ and } y \in \mathrm{FV}(N).
\end{array}
$$

(In 1.12(e)–(g), $y \not\equiv x$; and in (g), z is chosen to be the first variable $\notin \mathrm{FV}(NP)$.)

Remark 1.13 The purpose of clause 1.12(g) is to prevent the intuitive meaning of $[N/x](\lambda y.P)$ from depending on the bound variable y. For example, take three distinct variables w, x, y and look at

$$[w/x](\lambda y.x).$$

The term $\lambda y.x$ represents the constant function whose value is always x, so we should intuitively expect $[w/x](\lambda y.x)$ to represent the constant function whose value is always w. And this is what we get; by 1.12(f) and (a) we have

$$[w/x](\lambda y.x) \;\equiv\; \lambda y.w.$$

Now consider $[w/x](\lambda w.x)$. The term $\lambda w.x$ represents the constant function whose value is x, just as $\lambda y.x$ did. So we should hope that $[w/x](\lambda w.x)$ would represent the constant function whose value is always w.

But if $[w/x](\lambda w.x)$ was evaluated by (f), our hope would fail; we would have

$$[w/x](\lambda w.x) \;\equiv\; \lambda w.w,$$

which represents the identity function, not a constant function. Clause (g) rescues our hope. By (g) with $N \equiv y \equiv w$, we have

$$\begin{array}{rcll} [w/x](\lambda w.x) & \equiv & \lambda z.[w/x][z/w]x & \\ & \equiv & \lambda z.[w/x]x & \text{by 1.12 (b)} \\ & \equiv & \lambda z.w, & \end{array}$$

which represents the desired constant function.

Exercise 1.14 * Evaluate the following substitutions:

(a) $[(uv)/x]\ (\lambda y.x(\lambda w.vwx))$,

(b) $[(\lambda y.xy)/x]\ (\lambda y.x(\lambda x.x))$,

(c) $[(\lambda y.vy)/x]\ (y\,(\lambda v.xv))$,

(d) $[(uv)/x]\ (\lambda x.zy)$.

Lemma 1.15 *For all terms M, N and variables x:*

(a) $[x/x]M \equiv M$;

(b) $x \notin \mathrm{FV}(M) \implies [N/x]M \equiv M$;

(c) $x \in \mathrm{FV}(M) \implies \mathrm{FV}([N/x]M) = \mathrm{FV}(N) \cup \big(\mathrm{FV}(M) - \{x\}\big)$;

(d) $lgh([y/x]M) = lgh(M)$.

Proof Easy, by checking the clauses of Definition 1.12. □

Lemma 1.16 *Let x, y, v be distinct (the usual notation convention), and let no variable bound in M be free in vPQ. Then*

$$
\begin{array}{llll}
\text{(a)} & [P/v][v/x]M \equiv [P/x]M & & \textit{if } v \notin \mathrm{FV}(M); \\
\text{(b)} & [x/v][v/x]M \equiv M & & \textit{if } v \notin \mathrm{FV}(M); \\
\text{(c)} & [P/x][Q/y]M \equiv [([P/x]Q)/y][P/x]M & & \textit{if } y \notin \mathrm{FV}(P); \\
\text{(d)} & [P/x][Q/y]M \equiv [Q/y][P/x]M & & \textit{if } y \notin \mathrm{FV}(P),\ x \notin \mathrm{FV}(Q); \\
\text{(e)} & [P/x][Q/x]M \equiv [([P/x]Q)/x]M. & &
\end{array}
$$

Proof The restriction on variables bound in M ensures that 1.12(g) is never used in the substitutions. Parts (a), (c), (e) are proved by straightforward but boring inductions on M. Part (b) follows from (a) and 1.15(a), and (d) follows from (c) and 1.15(b). □

Definition 1.17 (Change of bound variables, congruence) Let a term P contain an occurrence of $\lambda x.M$, and let $y \notin \mathrm{FV}(M)$. The act of replacing this $\lambda x.M$ by

$$\lambda y.[y/x]M$$

is called a *change of bound variable* or an *α-conversion* in P. Iff P can be changed to Q by a finite (perhaps empty) series of changes of bound variables, we shall say *P is congruent to Q*, or *P α-converts to Q*, or

$$P \ \equiv_\alpha \ Q.$$

Example 1.18

$$
\begin{array}{rcl}
\lambda xy.x(xy) & \equiv & \lambda x.(\lambda y.x(xy)) \\
 & \equiv_\alpha & \lambda x.(\lambda v.x(xv)) \\
 & \equiv_\alpha & \lambda u.(\lambda v.u(uv)) \\
 & \equiv & \lambda uv.u(uv).
\end{array}
$$

Definition 1.17 comes from [CF58, p. 91]. The name 'α-converts' comes from the same book, as do other Greek-letter names that will be used later; some will look rather arbitrary but they have become standard notation.

Lemma 1.19

(a) *If $P \equiv_\alpha Q$ then* $\mathrm{FV}(P) = \mathrm{FV}(Q)$*;*

(b) *The relation $\equiv_\alpha$ is reflexive, transitive and symmetric. That is, for all P, Q, R, we have:*

$$\begin{array}{ll} \textit{(reflexivity)} & P \equiv_\alpha P, \\ \textit{(transitivity)} & P \equiv_\alpha Q,\ Q \equiv_\alpha R \implies P \equiv_\alpha R, \\ \textit{(symmetry)} & P \equiv_\alpha Q \implies Q \equiv_\alpha P. \end{array}$$

Proof For (a), see A1.5(f) in Appendix A1. For (b): reflexivity and transitivity are obvious; for symmetry, if P goes to Q by a change of bound variable, further changes can be found that bring Q back to P; details are in Appendix A1, A1.5(e). □

Lemma 1.20 *If we remove from Lemma 1.16 the condition on variables bound in M, and replace '$\equiv$' by '$\equiv_\alpha$', that lemma stays true.*

Proof By [CF58, p. 95, Section 3E Theorem 2(c)]. □

Lemma 1.21 $M \equiv_\alpha M'$, $N \equiv_\alpha N' \implies [N/x]M \equiv_\alpha [N'/x]M'$.

Proof By Appendix A1's Lemma A1.10. □

Note 1.22 Lemma 1.21 can be viewed as saying that the operation of substitution is well-behaved with respect to α-conversion: if we α-convert the inputs of a substitution, then the output will not change by anything more complicated than $\equiv_\alpha$. All the operations to be introduced later will also be well-behaved in a similar sense. (More details are in Appendix A1.) Thus, when a bound variable in a term P threatens to cause some trouble, for example by making a particular substitution complicated, we can simply change it to a new harmless variable and use the resulting new term instead of P.

Further, two α-convertible terms play identical roles in nearly all applications of λ-calculus, and are always given identical interpretations; so it makes sense to think of them as identical. In fact most writers use '$P \equiv Q$' to mean '$P \equiv_\alpha Q$'; the present book will do the same from Chapter 3 onward.

Remark 1.23 (Simultaneous substitution) It is possible to modify the definition of $[N/x]M$ in 1.12 to define *simultaneous substitution*

$$[N_1/x_1, \ldots, N_n/x_n]M$$

for $n \geq 2$. We shall not need the details here, as only very simple special