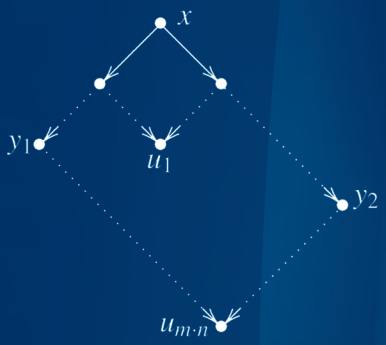
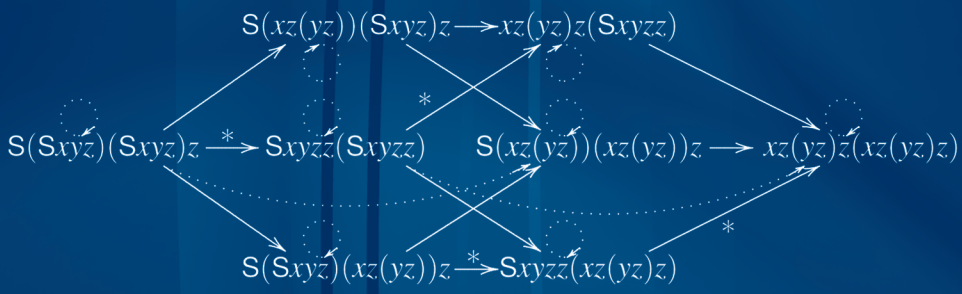
DISCRETE MATHEMATICS AND ITS APPLICATIONS
Series Editor KENNETH H. ROSEN

COMBINATORY LOGIC
PURE, APPLIED AND TYPED





Katalin Bimbó

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

COMBINATORY LOGIC
PURE, APPLIED AND TYPED

DISCRETE MATHEMATICS AND ITS APPLICATIONS

Series Editor
Kenneth H. Rosen, Ph.D.

R. B. J. T. Allenby and Alan Slomson, How to Count: An Introduction to Combinatorics, Third Edition

Juergen Bierbrauer, Introduction to Coding Theory

Katalin Bimbó, Combinatory Logic: Pure, Applied and Typed

Donald Bindner and Martin Erickson, A Student's Guide to the Study, Practice, and Tools of Modern Mathematics

Francine Blanchet-Sadri, Algorithmic Combinatorics on Partial Words

Richard A. Brualdi and Dragoš Cvetković, A Combinatorial Approach to Matrix Theory and Its Applications

Kun-Mao Chao and Bang Ye Wu, Spanning Trees and Optimization Problems

Charalambos A. Charalambides, Enumerative Combinatorics

Gary Chartrand and Ping Zhang, Chromatic Graph Theory

Henri Cohen, Gerhard Frey, et al., Handbook of Elliptic and Hyperelliptic Curve Cryptography

Charles J. Colbourn and Jeffrey H. Dinitz, Handbook of Combinatorial Designs, Second Edition

Martin Erickson, Pearls of Discrete Mathematics

Martin Erickson and Anthony Vazzana, Introduction to Number Theory

Steven Furino, Ying Miao, and Jianxing Yin, Frames and Resolvable Designs: Uses, Constructions, and Existence

Mark S. Gockenbach, Finite-Dimensional Linear Algebra

Randy Goldberg and Lance Riek, A Practical Handbook of Speech Coders

Jacob E. Goodman and Joseph O'Rourke, Handbook of Discrete and Computational Geometry, Second Edition

Jonathan L. Gross, Combinatorial Methods with Computer Applications

Jonathan L. Gross and Jay Yellen, Graph Theory and Its Applications, Second Edition

Titles (continued)

Carlos J. Moreno and Samuel S. Wagstaff, Jr., Sums of Squares of Integers

Dingyi Pei, Authentication Codes and Combinatorial Designs

Kenneth H. Rosen, Handbook of Discrete and Combinatorial Mathematics

Douglas R. Shier and K.T. Wallenius, Applied Mathematical Modeling: A Multidisciplinary Approach

Alexander Stanoyevitch, Introduction to Cryptography with Mathematical Foundations and Computer Implementations

Jörn Steuding, Diophantine Analysis

Douglas R. Stinson, Cryptography: Theory and Practice, Third Edition

Roberto Togneri and Christopher J. deSilva, Fundamentals of Information Theory and Coding Design

W. D. Wallis, Introduction to Combinatorial Designs, Second Edition

W. D. Wallis and J. C. George, Introduction to Combinatorics

Lawrence C. Washington, Elliptic Curves: Number Theory and Cryptography, Second Edition

DISCRETE MATHEMATICS AND ITS APPLICATIONS
Series Editor KENNETH H. ROSEN

COMBINATORY LOGIC
PURE, APPLIED AND TYPED

Katalin Bimbó
University of Alberta
Edmonton, Canada

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

Titles (continued)

Jonathan L. Gross and Jay Yellen, Handbook of Graph Theory

David S. Gunderson, Handbook of Mathematical Induction: Theory and Applications

Richard Hammack, Wilfried Imrich, and Sandi Klavžar, Handbook of Product Graphs, Second Edition

Darrel R. Hankerson, Greg A. Harris, and Peter D. Johnson, Introduction to Information Theory and Data Compression, Second Edition

Darel W. Hardy, Fred Richman, and Carol L. Walker, Applied Algebra: Codes, Ciphers, and Discrete Algorithms, Second Edition

Daryl D. Harms, Miroslav Kraetzl, Charles J. Colbourn, and John S. Devitt, Network Reliability: Experiments with a Symbolic Algebra Environment

Silvia Heubach and Toufik Mansour, Combinatorics of Compositions and Words

Leslie Hogben, Handbook of Linear Algebra

Derek F. Holt with Bettina Eick and Eamonn A. O'Brien, Handbook of Computational Group Theory

David M. Jackson and Terry I. Visentin, An Atlas of Smaller Maps in Orientable and Nonorientable Surfaces

Richard E. Klima, Neil P. Sigmon, and Ernest L. Stitzinger, Applications of Abstract Algebra with Maple™ and MATLAB®, Second Edition

Patrick Knupp and Kambiz Salari, Verification of Computer Codes in Computational Science and Engineering

William Kocay and Donald L. Kreher, Graphs, Algorithms, and Optimization

Donald L. Kreher and Douglas R. Stinson, Combinatorial Algorithms: Generation Enumeration and Search

Hang T. Lau, A Java Library of Graph Algorithms and Optimization

C. C. Lindner and C. A. Rodger, Design Theory, Second Edition

Nicholas A. Loehr, Bijective Combinatorics

Alasdair McAndrew, Introduction to Cryptography with Open-Source Software

Elliott Mendelson, Introduction to Mathematical Logic, Fifth Edition

Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, Handbook of Applied Cryptography

Richard A. Mollin, Advanced Number Theory with Applications

Richard A. Mollin, Algebraic Number Theory, Second Edition

Richard A. Mollin, Codes: The Guide to Secrecy from Ancient to Modern Times

Richard A. Mollin, Fundamental Number Theory with Applications, Second Edition

Richard A. Mollin, An Introduction to Cryptography, Second Edition

Richard A. Mollin, Quadratics

Richard A. Mollin, RSA and Public-Key Cryptography

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20110621

International Standard Book Number-13: 978-1-4398-0001-0 (eBook - PDF)

Contents

Preface

Combinatory logic was invented in the 1920s and has been developing ever since — often in the company of λ-calculi. Both of these formalisms, together with their variants, can capture the notion of a computable function or algorithm. This places combinatory logic next to recursion theory and computer science. On the other hand, typed combinatory logic straightforwardly connects with intuitionistic logic and other nonclassical logics, for example, relevance logics and linear logic. All this links combinatory logic to mathematical logic, philosophical logic and computational logic.

This book intends to present combinatory logic — starting with the basics — as a self-contained subject. λ-calculi are mentioned, because it seems unavoidable due to the historical development of these areas. However, λ-calculi are not given here as much prominence as is some other texts.

The key themes that are explicit (and sometimes implicit) in the following chapters are the connections to *computability* and to *nonclassical logics*. This implies that some other matters are given less emphasis. Nevertheless, the book aims to give a snapshot of the current state of the development of combinatory logic — together with pointers toward further possible advancements.

There is a comprehensive *two-volume book* on combinatory logic by Haskell B. Curry and his collaborators and students. Those volumes, however, had been completed by the late 1950s and early 1970s, respectively. Of course, mathematical and logical knowledge is typically cumulative, but the way concepts and results are presented might change even when the core results are preserved. Arguably, the role and the perception of combinatory logic within the discipline of logic has changed in the past 30–40 years. During those decades, many new results have been obtained. This creates a need and imparts a justification for a new presentation of the subject with some emphases shifted.

Combinatory logic is briefly mentioned or dealt with in quite a few *books on* λ*-calculi*, because of the relationship between λ-calculi and combinatory logics. (See, for instance, [10], [79] and [87].) I take the reverse approach here: the book focuses on combinatory logic itself, and I give a concise introduction to λ-calculi (in chapter 4), primarily, for the sake of comparison.

Curry seems to have deemed important to supplement his books with sections on history. (See, for instance, [53], [52] and [54].) Combinatory logic has *matured* since its early days, and it seems to me that brief sections would not suffice to provide an accurate historical account. Recently, separate and detailed publications appeared on the history of combinatory logic and λ-calculus in volumes on the history of logic, which also suggest that the history of the subject is better to be treated in itself.

In sum, I did not aim at providing a historical account of the development of combinatory logic in this book. Instead of trying to follow previous presentations closely, for the sake of historical accuracy, I intended to give an exposition in which the concepts are introduced as the results require them. Although some of the results are labeled by their usual labels, such as "Church–Rosser theorem," I did not try to attach a label to each claim, lemma and theorem in order to point to the first or to a widely accessible publication. An attempt to conduct more vigorous labeling either would have limited the scope of the presentation or would have opened the door for misattributions due to minor (or even more significant) differences in the underlying concepts. An extensive *bibliography* is included at the end of the book, which also includes texts that are either exclusively historical or at least more historically oriented than this book.

I wrote this book to give *a state-of-the-art view of combinatory logic* in the early 21st century. In order to make the book relatively easy to read, I started with an elementary introduction and I placed into the appendix various concepts and definitions that are useful or occasionally presupposed in the main text of the book, but do not belong to combinatory logic itself. I hope that this will help anybody to read the book, who has some aptitude toward formal thinking.

Combinatory logic is not as widely taught as I think it should be. Therefore, a purpose of this book is to serve as a readily accessible source of knowledge for experts in logic, computer science, mathematics, philosophy and certain related disciplines, who are perhaps, less familiar with this branch of logic.

Another potential use of the book, what I kept in mind during the writing, is as a text in a course. The examples and exercises in the text are intended to facilitate learning combinatory logic, alone or in the context of a course. (The starring of the exercises indicates the expected difficulty of their solutions.) The first couple of chapters of the book (perhaps, with a selection of further sections from later chapters) covers topics that can provide the content of an undergraduate course on combinatory logic. The whole book is suitable to be used as the main text for a graduate course — at a beginning or advanced level, depending on the background of the students. Because of the many connections combinatory logic has, I hope that the book will also be used to supplement courses, for example, on philosophical logic, on computability and on the foundations of mathematics.

My first acquaintance with combinatory logic happened in the early 1990s, when I took a course of Raymond Smullyan, at Indiana University in Bloomington. The power and elegance of combinatory logic quickly enthralled me.

The close connection between nonclassical logics and combinatory logic reinforced my interest in the subject. Among relevance logicians, it is common knowledge that Alonzo Church invented the implicational fragment of the logic of relevant implication (what he called "the theory of weak implication"), which corresponds to types for his $\lambda\mathsf{I}$-calculus. J. Michael Dunn and Robert K. Meyer invented structurally free logics and dual combinators in the 1990s. These logics both tighten the

connection between relevance logic and combinatory logic, and expand the bond between combinatory logic and a wide range of nonclassical logics.

My own research results in combinatory logic concern some of the newer developments such as dual combinatory logic. The connection between nonclassical logics and combinatory logic means that I always keep in mind combinators when I work on a substructural logic. Some of my work and results in the proof theory of relevance logic (and indirectly in the proof theory of classical logic) had been motivated by combinatory logic and structurally free logics. The splice of combinatory logic and relevance logic has proved fruitful in many ways. For example, J. Michael Dunn and I have solved the famous problem of $T_{\rightarrow}$, which remained open for half a century, by combining insights from proof theory and combinatory logic.

I am indebted to Robert Stern, the acquiring editor for this book, for his patience and for allowing me to ask for a new deadline (or two) for the submission of the manuscript. (The project of writing this book was substantially delayed when — somewhat unexpectedly — I had to teach a course on real ethics, that is, on formal decision theory as part of my regular teaching duties at the University of Alberta.) I am also grateful to Jennifer Ahringer for her help during the production process.

I am grateful to Graham Sullivan, who, as a research assistant, proofread parts of the book and provided helpful comments and corrections. I am indebted to the University of Alberta for awarding a grant from the Endowment Fund for the Future Support for the Advancement of Scholarship Research Fund that allowed me to have a research assistant for a couple of months at the time when the manuscript of the book was nearing its completion.

The book was typeset using the program TEX (by D. Knuth) with the LATEX format. I also utilized various packages that were developed under the auspices of the American Mathematical Society.

The actual writing of this book took place in Edmonton. I always enjoy the prospect of having another snow fall, what can (and sometime does) happen here as early as September and as late as June.

Edmonton, 2011 KB

This page intentionally left blank

Chapter 1

Elements of combinatory logic

Functions are everywhere, and combinatory logic is one of the theories that make them their objects of study. To start with, let us consider a simple function such as $+$, the addition of integers. $+$ can be applied to numbers; for instance, $46+68$ is the result of this application, which is also denoted by 114. Then we can view $+$ as a certain map from ordered pairs of integers into integers, or, further, as *a set of ordered triples*. From another perspective, $+$ can be thought of as *a rule to compute* the sum of two integers — indeed, this is how each of us first became acquainted with $+$. In both cases, knowing that $+$ is commutative and associative (and has other properties) is useful in identifying seemingly different versions of $+$. Just as light has a dual character — being a wave and being a beam of particles — functions have two sides to them: they are collections of input–output pairs, and they are rules for computation from inputs to the output.

Combinatory logic (henceforth, CL) is the logic of *functions*, in which the two aspects of functions mentioned above are in constant interplay. Functions are the objects in CL, and every object of CL can be thought of as a function. The functions need not always be thought of as functions on numbers, though we will see in chapter 3 how to represent numerical functions in CL.

1.1 Objects, combinators and terms

The objects in CL are called *terms*, and they stand for functions. Some of the terms cannot be divided into more elemental components: such terms are either *constants* or *variables*. The only way to put terms together is via (function) *application*, and the result of this operation is always a term.

Let us consider some examples from elementary algebra before proceeding to formal definitions. The natural numbers are denoted, in the usual decimal notation, by nonempty strings of digits like $0, 1, \ldots, 11, \ldots, 98\,765, \ldots$. (In formalized arithmetic, these numbers are denoted, respectively, by 0 and the successor function applied sufficiently many times to 0.) These expressions are constants, although in an elementary exposition they are not thought of as functions. Another kind of constants are functions like $+$ (addition), $\cdot$ (multiplication), $-$ (subtraction), etc. The three functions we listed are *binary* (i.e., each takes two arguments), but there are func-

tions of other *arities* too. The absolute value function has arity 1, that is, it is *unary*, and we can have functions taking n arguments (where n is a positive integer). It is just not that useful in practice to have a *quinary addition*, let us say $+_5$, that produces the sum of five numbers. (A quinary addition is easily emulated by the binary addition, because $+$ is associative.) Variables may enter into algebraic expressions too, and they are often denoted by $x, y, z, x_0, x_1, x_2, \ldots$.

So far we emphasized some *similarities* between algebraic terms and combinatory terms. They both may contain constants and variables. Each function takes a fixed number of inputs (i.e., arguments), but different functions can have more or fewer arguments than others. Also, the use of delimiters (such as parentheses) is common. $5+7\cdot 13$ is ambiguous — unless we have agreed that $\cdot$ takes precedence over $+$. In other words, $(5+7)\cdot 13$ and $5+(7\cdot 13)$ are different terms; moreover, their value is different too: $(5+7)\cdot 13$ is 156, whereas $5+(7\cdot 13)$ is 96.

A notable *difference* between algebraic and combinatory terms is that in elementary algebra functions such as $+$ are applicable to numbers only. That is, we would dismiss as nonsensical (or as "a misprint") something like $+\cdot(5+\cdot)$, where $\cdot$ seems to have $+$ as its argument and vice versa. We do not make any similar assumptions about the nonapplicability of terms in CL. We may reiterate also that variables in CL may stand for functions. This happens much more infrequently in algebra, except perhaps in definitions of properties of functions. For instance, $f(x,y)$ is called a commutative operation when $f(x,y)$ equals $f(y,x)$, for all x and y. This definition applies to *any* binary operation; that is, the whole sentence is understood as if "for all f" were prefixed to it. Lastly, function application is a binary operation in CL that is explicitly introduced (though typically suppressed in the notation). The more explicit and more detailed analysis of function application in CL leads to several notions of computing with terms.

We assume that the language of CL contains *constants*, typically, S and K. However, it may contain other constants too such as B, C, W, I and M — to name some. These constants are called *combinators*. There might be other sorts of constants in the language, including constants that are not definable from S and K (in a sense that we will explain later). B, C, W, M and I are all definable from S and K. For the definition of the set of terms, it is not particularly interesting which constants are included in the language, except that the definition, of course, applies only to those constants that are included. (We focus, especially in the first half of the book, exclusively on combinators.) We also assume that there is a denumerable set of *variables*, that is, there are infinitely many variables, but countably many (i.e., $\aleph_0$ many). The language includes a binary operation called *function application* that is denoted by juxtaposition.

DEFINITION 1.1.1. The *set of* CL*-terms* (or terms, for short) is inductively defined by (1)–(3).[1]

(1) If Z is a constant, then Z is a term;

[1] As usual, we leave tacit and hence omit mentioning that the least set generated by clauses (1)–(3) is the set, that is defined. This will be our practice in what follows.

(2) if x is a variable, then x is a term;

(3) if M and N are terms, then so is (MN).

We use uppercase sans serif letters for combinators. The letters chosen for particular combinators are often standard. To refer ambiguously to some combinator, we use Z; in other words, Z is a meta-variable for combinators.[2] The lowercase letters, $x, y, z, \ldots$, with or without indices, serve as variables (as well as meta-variables for variables). The uppercase letters M, N, P and Q, with or without indices, are meta-variables for CL-terms in general. To put it differently, M can be instantiated with x or I, but also with $(\mathsf{S}(x(xy)))$, etc. The binary application operation figures into the term (MN) in clause (3). We do not add a symbol like $\cdot$ or $+$ to the term that results from M and N, but the resulting term is enclosed in parentheses.

Example 1.1.2. S and K are terms, and so are the variables standing alone. These terms are called *atomic* to distinguish them from the rest of the terms, which are called *complex* or *compound*. Some of the latter are $(\mathsf{I}x)$, $(x(yz))$, $(z\mathsf{S})$ and $((\mathsf{SI})\mathsf{I})$.

There are denumerably many atomic terms and there are denumerably many complex terms; in total, the set of CL-terms is *denumerable* (i.e., the cardinality of the set of CL-terms is $\aleph_0$).

Exercise 1.1.1. Prove informally (or prove using structural induction) that every CL-term either is atomic or starts with a '('.

Exercise 1.1.2. Sort the terms listed into the following categories: "atomic term," "complex term" and "nonterm" (i.e., an expression that is not a term). (x), SKK, x_{12}, $(xx))(((xx)x)x)$, $(((\mathsf{S}(\mathsf{KS}))\mathsf{K})(x(yz)))$, $(((\mathsf{BM})((\mathsf{BW})\mathsf{B}))x)$, $((((\mathsf{W}M)N)N)\mathsf{W})$, C, z, $(((x_1\mathsf{S})((x_2\mathsf{S})x_3))((x_5(x_4\mathsf{K}))y))$.

Exercise*1.1.3. Prove informally (or by structural induction) for every term that if the term contains parentheses, then the parentheses are well-balanced. (The latter means that there are equal numbers of '('s and ')'s, and that scanning the term from left to right, there are never more right parentheses that left ones.)

Exercise 1.1.4. Prove that not all expressions that contain well-balanced parentheses are CL-terms.

We will often use *meta-terms* instead of terms, that is, expressions that contain some of the meta-variables M, N, P, ... By those (meta-)terms, we refer to an arbitrary term that instantiates the expression.

Another notational convention concerns *omitting parentheses*. Delimiters pile up quickly, and then terms soon become practically unreadable. In CL, parentheses are normally omitted from *left-associated* terms, that is, a term of the form $((MN)P)$

[2] Z is a schematic letter that stands for any of the combinators in the meta-language, that is, in the language we use to talk *about* CL. We use 'meta-variable' similarly for other types of expressions.

may be written as (MNP). The outmost parentheses are, as a rule, omitted too; continuing the example, we get MNP.[3]

Example 1.1.3. $(\mathsf{B}xy)z$ is shorthand for $(((\mathsf{B}x)y)z)$. $\mathsf{BWB}x(\mathsf{BWB}x)$ is a result of omitting some parentheses from $((((\mathsf{BW})\mathsf{B})x)(((\mathsf{BW})\mathsf{B})x))$. The former shorthand term could be further abbreviated by omitting the remaining pair of parentheses. The latter, however, cannot be further simplified.

There are terms from which parentheses may be omitted in various ways; hence, we may get more than one abbreviated term. On the other hand, the insertion of all omitted parentheses leads to a unique term, which is important, because we do not want to introduce ambiguity into our notation. In view of this remark, we will call the abbreviated terms simply "terms" — unless the distinction between them is the point, as in the next couple of exercises.

Exercise 1.1.5. Restore all the parentheses in the following abbreviated terms.
$\mathsf{K}x(\mathsf{W}yz)y$, xxx, $(\mathsf{SI})y\mathsf{B}$, $x_{14}x_{152}$, $M_1(M_2(M_3xyy)z)$.

Exercise 1.1.6. Omit as many parentheses as possible from the following terms (without introducing ambiguity). $(((yz)(\mathsf{II}))x)$, $((\mathsf{M}x)((\mathsf{B}y)(\mathsf{W}z)))$, $(\mathsf{W}(\mathsf{I}x_{145}(\mathsf{I}x_{72}))x_{58})$, $(((\mathsf{SK})\mathsf{K})x)$, $(((\mathsf{S}M)M)(\mathsf{I}(NP)))$.

Exercise 1.1.7. Is there a term that has at least two (distinct) abbreviated forms such that neither can be obtained from the other by omitting parentheses? (Hint: If the answer is "yes," give an example; if the answer is "no," then prove that such abbreviated terms cannot exist.)

Exercise 1.1.8. Prove informally (or by structural induction) that all terms contain an even number of well-balanced parentheses, and that omitting parentheses preserves this property.

Exercise 1.1.9. Give an example of an expression (from the language of CL) that is not a term or an abbreviated term, but has an even number of parentheses. Give an example of an expression (from the language of CL) that is not a term or an abbreviated term, but contains well-balanced parentheses.

Now that we have a precise definition of CL-terms, it will be helpful to point out a few more similarities and differences between algebraic terms and CL-terms. Arithmetic functions such as multiplication and addition are binary, and the function symbol is usually placed *between* the two arguments. We will see in the next section that the arity of W is 2; nonetheless, the arguments of W *follow* the function symbol (that is, the arguments come after W). In general, putting a function (or relation) symbol between its arguments is called *infix* notation. In CL, a *prefix* notation is used instead, that is, functions precede (or are prefixed to) their arguments.

[3] We parenthesize M, N, P, ..., as if they were atomic terms — as long as they have not been instantiated. This, of course, does *not* imply that they have to be replaced or instantiated by atomic terms.

In mathematics, in general, there is great variation as to where the arguments of a function are located. Consider as examples exponentiation, logarithmic functions, integrals and derivatives. For less frequently used functions, the notation is less idiosyncratic: an n-ary function may be simply denoted by $f(x_1,\dots,x_n)$. If f were a constant in CL, then the term in which the function f is followed by the arguments $x_1,\dots,x_n$ would look like $(\dots(fx_1)\dots x_n)$. In other words, the commas are all dropped and a pair of parentheses surrounds a function applied to an argument rather than a series of arguments.

Exercise 1.1.10. Assume that # and $*$ are combinatory constants (of arity 2) that stand, respectively, for addition and multiplication in CL. (a) Translate the following arithmetical terms into CL. $z+z$, $x\cdot y$, $(x+y)\cdot x$, $x_1\cdot(x_2\cdot x_3\cdot x_3+x_4\cdot x_5+x_7)$, $x\cdot(x+y)$. (b) Translate the following terms from CL into arithmetics. $\#x(\#xx)$, $(*(\#yz))((\#x)y)$, $*(*(*(*xz)y)y)x)(yx)$, $\#M(N\#P)$, $\#\#(*\mathsf{KI})(*\mathsf{K}*)$.

We have not exhausted the range of investigations of syntactic properties of terms.[4] However, we introduce here yet another syntactic concept. Informally, M is a subterm of N if M is itself a term and M is part of N.

DEFINITION 1.1.4. The *subterm of* relation on the set of CL-terms (i.e., M is a subterm of N) is defined inductively by (1)–(5).

(1) x is a subterm of x;

(2) Z is a subterm of Z;

(3) M is a subterm of the terms (NM) and (MN);

(4) (NP) is a subterm of (NP);

(5) if M is a subterm of N and N is a subterm of P, then M is a subterm of P.

All subterms of a term are themselves terms. Terms have more that one subterm — unless they are atomic; and all terms are subterms of infinitely many terms. The joint effect of (1), (2) and (4) is that every term is a subterm of itself, which may appear strange at first (given the ordinary meaning of "sub-"). However, the *reflexivity* of the "subterm of" relation is intended, because it simplifies some other definitions. (The notion of a *proper subterm* is also definable — see section A.1.) Clause (5) means that the "subterm of" relation is *transitive*; indeed, it is a partial order on the set of terms.

Exercise 1.1.11. List all the subterms of each of the following terms. $(x(\mathsf{W}x)y)$, $yy(y(yy))$, $(\mathsf{S}(\mathsf{KW})\mathsf{S})(\mathsf{BWW})$, $z\mathsf{S}(y\mathsf{S})(x\mathsf{WSK})$, $PM((PN)(PP))M$, $\mathsf{W}Mx(NNPPy)$, $(\mathsf{B}M(x)(NM))$.

[4] Some further notions and definitions concerning terms, such as the definition of left-associated terms, the definition of occurrences of terms, the definition of free occurrences of a variable, etc., may be found in section A.1 of the appendix.

We stated but did not prove that the "subterm of" relation is a partial order. Granted that reflexivity and transitivity are established, it remains to show that the relation is *antisymmetric*. This is the content of the next exercise.

Exercise 1.1.12. Prove that if M is a subterm of N and N is a subterm of M, then M is the same term as N.

Exercise 1.1.13. Complex CL-terms have more than one term that is their subterm. Define inductively an operation that gives *the set of subterms* for a term.

Exercise 1.1.14. Exercise 1.1.11 in effect asked you to list the elements of the set of subterms of seven terms. Give another definition of the set of subterms of a term using the "subterm of" relation.

1.2 Various kinds of combinators

Combinators are characterized by *the number of arguments* they can have (i.e., their *arity*) and by *the effect* they have when applied to as many arguments as their arity. The arity of a combinator is always *a positive integer*. For example, I is a unary combinator, whereas S takes three arguments.

The effect of the application of a combinator is given by its *axiom*. A combinatory axiom consists of two terms with a symbol (such as $\rhd$) inserted between them. The following table shows the axioms of some well-known combinators.[5]

$\mathsf{I}x \rhd x$	$\mathsf{B}xyz \rhd x(yz)$	$\mathsf{S}xyz \rhd xz(yz)$
$\mathsf{K}xy \rhd x$	$\mathsf{C}xyz \rhd xzy$	$\mathsf{W}xy \rhd xyy$
$\mathsf{M}x \rhd xx$	$\mathsf{B}'xyz \rhd y(xz)$	$\mathsf{J}xyzv \rhd xy(xvz)$

The terms on the left-hand side of the $\rhd$ tacitly indicate the arity of the combinator, because the combinator precedes as many variables as its arity. Notice also that these variables are not assumed to be the same. For instance, we have the term $\mathsf{W}xy$ (not $\mathsf{W}xx$) on the left-hand side of the $\rhd$ on the second line.

The term on the right-hand side in an axiom shows the term that results after the application of the combinator. In the resulting terms, some of the variables may be repeated, omitted, or moved around. All the combinators listed above (but not all combinators in CL) have the property that their application does not introduce new variables and the resulting term is built solely from some of the variables that appeared as their arguments. Such combinators are called *proper*. Some combinators that are *not proper* (i.e., *improper*), are extremely important, and we will return to them later. However, they are the exceptions among the improper combinators.

[5] B' is not a single letter, and this combinator is sometimes denoted by Q. B' is closely related to B; this explains the notation, which we retain because it is widely used.

Variables are used in logic when there is an operator binding them or there is a rule of substitution. CL contains *no variable binding operators*, which is one of the best features of CL; however, substitution is a rule. The combinatory axioms above gave a characterization of applications of combinators to variables only, but of course, we would like combinators to be applicable to arbitrary terms. *Substitution* ensures that this really happens, as we illustrate now.

Example 1.2.1. $\mathsf{I}(yz) \rhd yz$ is an *instance* of the axiom for I with yz substituted for x in both terms (that is, in $\mathsf{I}x$ and in x) in the axiom. Another instance of the same axiom is $\mathsf{I}(y(xz)) \rhd y(xz)$. (Notice that the presence of x in the term that is being substituted is unproblematic, and there is no danger of "circularity" here.)

We get yet another example by substituting I for x, M for y and z for z (or leaving z as it is) in the axiom for B. $\mathsf{BIM}z$ is the left-hand side term, and $\mathsf{I}(\mathsf{M}z)$ is the right-hand side term. The latter term can be viewed also as obtained from the left-hand side term in the axiom for I by substituting $\mathsf{M}z$ for x, which then gives $\mathsf{M}z$ on the right.

The last example shows that we may be interested in successive applications of combinators to terms. It also shows that substitution in distinct terms may result in the same term. Furthermore, the same term may be substituted for different variables; hence, the use of distinct variables in the axioms is not a restriction at all, but rather a way to ensure generality. The informal notion of substitution may be made precise as follows.

DEFINITION 1.2.2. (SUBSTITUTION) The *substitution* of a term M for the variable x in the term N is denoted by N^M_x, and it is inductively defined by (1)–(4).

(1) If N is x, then N^M_x is M, (i.e., x^M_x is M);

(2) if N is y and y is a variable distinct from x, then N^M_x is N, (i.e., y^M_x is y);

(3) if N is Z, then N^M_x is N, (i.e., Z^M_x is Z);

(4) if N is (P_1P_2), then N^M_x is $(P_1{}^M_x P_2{}^M_x)$, (i.e., $(P_1P_2)^M_x$ is $(P_1{}^M_x P_2{}^M_x)$).

Substitution is a *total operation* in CL. There is no restriction on the shape of N or on occurrences of x in N in the definition, though the result of the substitution, in general, will depend on what N looks like. Substitution can be summed up informally by saying that all the x's in N (if there are any) are turned into M's.

Substitution is a much more complicated operation in systems that contain a variable binding operator — such as λ-calculi or logics with quantifiers. Substitution was not very well understood in the early 20th century, and one of the motivations for combinatory logic — especially, in Curry's work in the 1930s — was the clarification of substitution. Another motivation — mainly behind Schönfinkel's work — was the elimination of bound variables from first-order logic. These two problems turn out to be one and the same, and the general solution in both cases leads to a *combinatorially complete* set of combinators. (We return to combinatorial completeness in the next section; see in particular definition 1.3.8 and lemma 1.3.9.)

Sometimes, it is convenient to substitute at once for more than one variable in a term. This operation is called *simultaneous substitution*. A simultaneous substitution can always be emulated by finitely many single substitutions, which means that simultaneous substitutions can be viewed as convenient notational devices. (However, see section A.1 for a more formal view of this operation.) We limit ourselves to a few examples.

Example 1.2.3. $(\mathsf{S}xyz)^{\mathsf{I},yy,\mathsf{J}}_{x,y,z}$ is $\mathsf{SI}(yy)\mathsf{J}$, that is, I is substituted for x, yy for y and J for z. A more revealing case is when one of the substituted terms contains a variable that is one of the variables for which a term is substituted simultaneously. For instance, $(\mathsf{J}xyzv)^{yy,\mathsf{W}}_{x,y}$ is the term $\mathsf{J}(yy)\mathsf{W}zv$ (rather than $\mathsf{J}(\mathsf{WW})\mathsf{W}zv$).

Other notations for the substitution of M for x include $[x/M]$ (and $[M/x]$) placed in front of or after the term on which the operation is performed. For instance, $[x/M]N$ or $[x_1/M_1, x_2/M_2]N$ are such notations, which may be clearer than N^M_x and $N^{M_1,M_2}_{x_1,x_2}$.[6] (We will use the "slash-prefix" notation too.)

Exercise 1.2.1. Write out the details of the following substitutions step-by-step. (a) $(\mathsf{I}x(\mathsf{W}x))^z_x$, (b) $[z/x(xy)]\mathsf{S}(xx)(yy)(xy)$, (c) $[x_1/\mathsf{M}, x_3/\mathsf{B}, x_4/\mathsf{W}](x_3x_1(x_3x_4x_3))$, (d) $(\mathsf{S}xyz)^{\mathsf{K},\mathsf{K},x}_{x,y,z}$, (e) $(\mathsf{C}xyM)^{PPN}_y$.

Substitution affects *all occurrences* of a *variable*. Another operation on terms affects *selected occurrences* of a *subterm* in a term, and it is called *replacement*. The result of replacing some subterms of a term by a term is, of course, a term. For our purposes, the visual identification of an occurrence of a subterm is sufficient — together with the remark that a completely rigorous numbering or identification of the subterms is possible.

Example 1.2.4. The result of the replacement of an occurrence of a subterm $\mathsf{SKK}x$ in $\mathsf{SKK}x$ by $\mathsf{I}x$ is $\mathsf{I}x$. (Recall that every term is its own subterm.) The result of replacement of the first occurrence (from left) of x in $\mathsf{KS}x(\mathsf{S}x)$ by M is $\mathsf{KS}M(\mathsf{S}x)$, whereas the replacement of the second occurrence of x yields $\mathsf{KS}x(\mathsf{S}M)$. This shows that *not all* occurrences of a subterm (even if that term happens to be a variable) need to be replaced. Lastly, the replacement of the second occurrence of BW in $\mathsf{BWBW}(\mathsf{BWBW})$ by SKC gives us $\mathsf{BWBW}(\mathsf{SKCBW})$. (This looks like, but in fact is not, a typo!)

In the first example, the term we started with and the term we ended up with are very closely related in a sense that will soon become clear. A similar relationship holds in the second example, but only accidentally — because of the occurrence of K in a specific spot in the term. The last two examples show no such relationship between the input and the output terms. Generally speaking, replacement induces

[6] There are still other ways to denote substitution. Occasionally, $\leftarrow$ or $\rightarrow$ is used instead of $/$. We do not intend to give an exhaustive list here, but you should be aware that the notation for substitution is anything but unvarying in the literature.

desirable relationships between input and output terms only if such relationships obtain between the replaced subterm and the newly inserted subterm.

The effect a combinator can have on a term may be thought of as *replacing* a term that is on the left-hand side of the $\rhd$ by a term that has the form of the term on the right-hand side of the $\rhd$. More precisely, such replacements can be performed *within* any terms according to instances of combinatory axioms.

The change that a combinator produces in a term is one of the most interesting criteria for classifying combinators. (We have already seen another grounds for categorization: the arity of combinators and the comparison of the set of atomic subterms of the terms on the left and on the right.)

Terms are "structured strings," that is, they typically contain parentheses that show grouping. Terms might differ with respect to the number of occurrences of some subterm, the order of subterms or whether a subterm occurs at all in a term.

There are *five salient features* that we use to categorize combinators.[7] The properties that are exemplified by I, B, K, C and M are the ones that are abstracted away and turned into defining properties of classes of combinators.

Identity combinators. The letter I is intended to remind us that I is a unary identity function. As such, the effect of an application of I is that we get back the input term. (To put it more scintillatingly, I simply "drops off" from the term.) Functions like I can be envisioned for any arity, and accordingly we call the n-ary combinator Z an *identity combinator* when its axiom is

$$(\ldots((\mathsf{Z}x_1)x_2)\ldots x_n) \rhd (\ldots(x_1x_2)\ldots x_n).$$

The binary identity combinator may be taken to be BI, because $\mathsf{BI}yz \rhd \mathsf{I}(yz)$ and $\mathsf{I}(yz) \rhd yz$ are instances of the axioms for B and I. (We already hinted at that we might consider series of replacement steps according to combinatory axioms; we will make this precise in the next section when we will define reduction to be a transitive relation.)

Associators. Recall that the only alteration B introduces into a term is association to the right. A term comprising two atoms, such as two variables x, y or two constants S, C, can be grouped only in one way. Hence, B has the least arity (3) among associators. An n-ary combinator Z is an *associator* when Z's axiom is

$$\mathsf{Z}x_1\ldots x_n \rhd M \text{ and } M \text{ is not a left-associated term.}$$

For example, M may be a term $(x_1\ldots x_n)$ with x_1 through x_n in the same order as on the left-hand side of the $\rhd$, but with at least one subterm of the form x_iN, where $i \neq 1$. The following Z_1 and Z_2 are both quinary associators.

$$\mathsf{Z}_1x_1x_2x_3x_4x_5 \rhd x_1x_2x_3(x_4x_5)$$

[7] We follow the usual classification of combinators (that may be found, e.g., in Curry and Feys [53]) with some slight modifications. A main difference is that we do not limit the use of labels such as "permutator" to regular combinators.

$$\mathsf{Z}_2 x_1 x_2 x_3 x_4 x_5 \rhd x_1 (x_2 x_3 x_4 x_5)$$

There are five possible ways to insert parentheses into a term that consists of four atomic terms. Curiously, B by itself is sufficient to define three combinators that yield the terms $x_1 x_2 (x_3 x_4)$, $x_1 (x_2 x_3 x_4)$ and $x_1 (x_2 (x_3 x_4))$ (when the arguments are x_1, x_2, x_3 and x_4 in that order), and an application of B yields $x_1 (x_2 x_3) x_4$. (Although B is a ternary combinator, we may form a term from the terms $\mathsf{B} x_1 x_2 x_3$ and x_4, that reduces to the desired term.) $\mathsf{B}(x_1 x_2) x_3 x_4 \rhd x_1 x_2 (x_3 x_4)$ is an instance of B's axiom, and so is $\mathsf{BB} x_1 x_2 \rhd \mathsf{B}(x_1 x_2)$. Then BB, sometimes denoted as D, associates x_3 and x_4 — leaving x_1 and x_2 grouped together.

$$\mathsf{D} x_1 x_2 x_3 x_4 \rhd x_1 x_2 (x_3 x_4)$$

Exercise 1.2.2. Find combinators Z_3 and Z_4 that are composed of B's and have axioms $\mathsf{Z}_3 x_1 x_2 x_3 x_4 \rhd x_1 (x_2 x_3 x_4)$ and $\mathsf{Z}_4 x_1 x_2 x_3 x_4 \rhd x_1 (x_2 (x_3 x_4))$.

Cancellators. An application of the combinator K to two arguments, let us say x and y, results in the term x, that is, y gets omitted or cancelled. This feature may seem spurious at first. However, when K and S are chosen as the only constants, the potential to form terms with some subterms disappearing after an application of K is essential. In general, an n-ary combinator Z is called a *cancellator* when the axiom of Z is

$$\mathsf{Z} x_1 \ldots x_n \rhd M \text{ with at least one of } x_1, \ldots, x_n \text{ having no occurrences in } M.$$

As another example of a cancellator, we could consider a combinator K^2 such that when it is applied to x and y the term y results. $\mathsf{K}\mathsf{I} x \rhd \mathsf{I}$ is an instance of the axiom for K; hence, $\mathsf{K}\mathsf{I}$ achieves the desired effect. We will see in chapter 3 that the capability to omit the first or the second argument that is exhibited by K and $\mathsf{K}\mathsf{I}$, as well as the definability of cancellators that retain only one of their n arguments is paramount, in the representation of recursive functions by combinators.

Permutators. The combinators C and S both change the order of some of their arguments. Recall that the right-hand side terms in their axioms are xzy and $xz(yz)$. In the former, y and z are swapped; in the latter, an occurrence of z precedes an occurrence of y. A combinator taking n arguments is a *permutator* when its axiom is of the form

$$\mathsf{Z} x_1 \ldots x_n \rhd M \text{ with } M \text{ containing an } x_j \text{ preceding an } x_i \text{ (for } 1 \leq i < j \leq n).$$

Of course, M may contain several occurrences of either variable, but in the definition we only require the existence of at least one pair of occurrences with x_j coming first.

Exercise 1.2.3. Find a ternary combinator (composed of the constants already introduced) such that if applied to x, y and z it produces the term yzx, that is, "it moves its first argument behind the two others." This combinator is usually denoted by R and its axiom is $\mathsf{R} xyz \rhd yzx$. The effect of R can be produced by combining the combinators that we already have.

Exercise 1.2.4. Recall that the axiom for the combinator B' is $\mathsf{B}'xyz \rhd y(xz)$. First, define the combinator B' using B and C. Next, find a combinator V that "reverses" the effect of R, that is, an application of V to three arguments x, y and z yields zxy. (Hint: B', B and some permutators may be useful in the latter definition.)

Duplicators. The combinators W and M "copy" or "duplicate" some of their arguments. W's application results in two occurrences of its second argument in the output term, and M returns two copies of its (only) input. An n-ary combinator Z is a *duplicator* when the axiom for Z is

$$\mathsf{Z}x_1 \ldots x_n \rhd M \text{ and } M \text{ contains more than one occurrence of an } x_i \ (1 \leq i \leq n).$$

Notice that S is not only a permutator, but a duplicator too, because $xz(yz)$ has two occurrences of z. S is an excellent example of a combinator that combines various kinds of effects. Indeed, in the terminology of [53], the last four categories would be called "combinators with such-and-such effect." (Cf. footnote 7.)

The only category of combinators in which the shape of the resulting term is completely specified by our definitions is that of identities. Combinators that integrate various effects are useful, but it is also helpful to be able to dissect their blended effect into components. This will become transparent when we will delineate a subset of the set of the terms that can model arithmetic functions.

Exercise 1.2.5. Classify all the combinators introduced so far according to their effects. (Hint: Some combinators will appear in more that one category.)

Exercise 1.2.6. Define or find combinators (among the already introduced ones) that demonstrate each pair of properties that can appear together. (Hint: You need 6 combinators in total.)

We quickly mention two other classes of combinators now.

Regular combinators. Combinators may be thought of as affecting the *arguments* of a function. Let us assume that f is a binary function; then fxy is the application of f to x and y. $\mathsf{C}fxy$ takes f, x and y as arguments and yields fyx, that is, f applied to its arguments in reversed order. This motivates singling out combinators that keep their first argument "in the left-most place." These combinators are called *regular*, and the axiom for a regular combinator Z is

$$\mathsf{Z}x_1 \ldots x_n \rhd x_1 M.$$

The combinators I, K, S, W, C, B, J and D are all regular, whereas B' and R are not. Therefore, it is obvious by now that putting together regular combinators may give a combinator that is not regular. Because of the lack of preservation of regularity under application, we will not place an emphasis on regular combinators — despite the fact that they have an easy to understand informal meaning.

Proper combinators. We already mentioned this kind of combinators. Z is a *proper* combinator if its axiom is

$$\mathsf{Z}x_1 \ldots x_n \rhd M \text{ and } M \text{ contains no variables other than } x_1, \ldots, x_n.$$

Informally speaking, combinators that introduce a new variable are hardly sensible (because an arbitrary term can be substituted for that variable). On the other hand, some combinators that introduce a *constant* make perfect sense and are important (even though they are not proper). The archetypical combinator of this kind is the *fixed point* combinator, often denoted by Y.[8] The axiom for Y is $\mathsf{Y}x \rhd x(\mathsf{Y}x)$. If x is a function, then $\mathsf{Y}x$ is its fixed point when the axiom is viewed from right to left (or $\rhd$ is thought of as some sort of equality). That is, by replacing $\mathsf{Y}x$ by z, x by f and $\rhd$ by $=$ we get $fz = z$.

Exercise* **1.2.7.** Find a term built from some of the combinators mentioned so far (save Y) that has the effect of Y.

We introduced some combinators as constants at the beginning of this section. These are the *primitive combinators* (also called *undefined combinators*) in our system. However, we have used the term "combinator" informally to refer to complex terms as well. We make this use official now. A complex term that is built entirely from (primitive) combinators is called a *combinator* too.

The solution of the previous exercise shows that combining primitive proper combinators does not always preserve the property of being proper. Indeed, it is quite easy to generate improper combinators this way.

Example 1.2.5. C and I are both proper combinators, but CII is not. This particular combinator is interesting in the context of typed systems, because its principal type schema is $((A \to A) \to B) \to B$ (i.e., *specialized assertion*), which is a theorem of classical propositional logic and a characteristic axiom of the logic of entailment.

1.3 Reductions and combinatory bases

The understanding of the structure of CL-terms as well as of some of their other syntactic features is vital to grasping CL. However, CL is more about *relations* between terms than about the terms themselves. We have used phrases such as "combinators cause changes in terms," and such changes may be characterized by *pairs of terms*: the first term is the term in which the combinator is applied and the second term is the term that results. We can describe the axioms this way too, and then define one-step reduction.

DEFINITION 1.3.1. If Z is an n-ary combinator, then a term of the form $\mathsf{Z}M_1 \dots M_n$ is a *redex*. The *head* of this redex is Z, and $M_1, \dots, M_n$ are its *arguments*.

A redex may contain one or more combinators, because the terms that instantiate the meta-variables $M_1, \dots, M_n$ may or may not have combinators and redexes in

[8] This combinator is sometimes labeled as "fixpoint" or "paradoxical" combinator. We hasten to point out that pure CL is consistent, and so the latter name might seem to cozen.

them. Also, Z may occur several times in the term, but it surely occurs at least once. That is, the definition is to be understood to mean that the particular occurrence of Z that is apparent in the definition of a redex is the head of the redex.

DEFINITION 1.3.2. (ONE-STEP REDUCTION) Let Z be a primitive combinator with axiom $\mathsf{Z}x_1 \ldots x_n \rhd P$. If N is a term with a subterm of the form $\mathsf{Z}M_1 \ldots M_n$, and N' is N with that subterm replaced by $[x_1/M_1, \ldots, x_n/M_n]P$, then N *one-step reduces* to N'. This is denoted by $N \rhd_1 N'$.

The definition is not as complicated as it may first seem. We already (tacitly) used $\rhd_1$ in examples to illustrate various types of combinators. When we said that $\mathsf{I}(yz)$ yields yz, we applied one-step reduction. $\mathsf{I}M$ is a subterm of $\mathsf{I}(yz)$ with M standing for yz. In I's axiom, x (on the right-hand side of the $\rhd$) plays P's role in the above definition, and $[x/yz]x$ is yz. Thus $\mathsf{I}(yz) \rhd_1 yz$ — it is really quite straightforward.

Example 1.3.3. The term $\mathsf{BWB}x(\mathsf{BWB}x)$ contains two redexes. The head of one of them is the first occurrence of B (counting from left to right); the head of the other is the third occurrence of B. Having chosen the second redex for one-step reduction, we get $\mathsf{BWB}x(\mathsf{BWB}x) \rhd_1 \mathsf{BWB}x(\mathsf{W}(\mathsf{B}x))$. The latter term contains one redex only, and after one-step reducing that, we get $\mathsf{W}(\mathsf{B}x)(\mathsf{W}(\mathsf{B}x))$. (This term also has a redex, because the one-step reduction created a new redex.)

Exercise* 1.3.1. Consider the term $\mathsf{S}(\mathsf{KS})\mathsf{KSKS}$. Perform as many one-step reductions as possible in every possible order. (Hint: Write out the one-step reductions as sequences of terms.)

One-step reduction may yield a term with no redexes, and it may yield a term with one or more remaining or new redexes. A one-step reduction of a duplicator may increase the number of redexes — including the number of occurrences of some already existing redexes. In other words, the number of successive one-step reductions counts the number of reduced redexes, but this number does not necessarily equal the number of redexes in the starting term minus the number of redexes in the resulting term. The term $\mathsf{BWB}x(\mathsf{BWB}x)$ (from example 1.3.3) has two redexes. Once the first redex from the left is reduced, the resulting term $\mathsf{W}(\mathsf{B}x)(\mathsf{BWB}x)$ contains a brand new redex headed by W.

Exercise 1.3.2. For each of the categories of combinators introduced in the previous section, consider terms in which one of the redexes M has a head belonging to that category, and all the other redexes are wholly inside arguments of the head of M. Work out how the number of occurrences of the already existing redexes changes with the one-step reduction of M.

Exercise* 1.3.3. Example 1.3.3 showed that one-step reduction may create *new redexes* — not merely *new occurrences* of already existing redexes. Can combinators from each category create new redexes? (Hint: If "yes," then describe the shapes of suitable terms before and after $\rhd_1$.)

The sample one-step reductions we performed on $\mathsf{BWB}x(\mathsf{BWB}x)$ show that redexes need not be reduced in the same order as they occur in a term from left to right. This suggests that we may define *reduction strategies* or *reduction orders* to have more control over the reduction process. One special order for reductions — proceeding from left to right — has a theoretical importance in connection to normal forms, as well as a practical significance in thinking about CL as modeling computation. Sequential programming languages force a linear order upon computational steps; hence, the implicitly nondeterministic view projected by definition 1.3.2 may not be always tenable.

One-step reductions surely entice us to contemplate successive one-step reductions. Indeed, given all the combinators with their diverse effects, what we are really interested in is the compound outcome of their repeated applications.

DEFINITION 1.3.4. (WEAK REDUCTION) The reflexive transitive closure of the one-step reduction relation is *weak reduction*, and it is denoted by $\rhd_{\mathrm{w}}$.

If R is a binary relation on a set A, then the *reflexive closure* of R, contains all pairs of elements of A of the form $\langle a,a\rangle$, for all $a \in A$. Accordingly, the weak reduction relation holds between every CL-term and itself: $M \rhd_{\mathrm{w}} M$. A term that has no redexes does not one-step reduce to any term, yet it weakly reduces to itself. Terms M such that $M \rhd_1 M$ are rare and exceptional. One example is MM.

If R is a binary relation on a set A, then R^+, the transitive closure of R includes all pairs $\langle a,c\rangle$ in which the second element is "accessible" from the first one via finitely many R-steps. If $Rab_1,\ldots,Rb_nc$ for some $b_1,\ldots,b_n$, then R^+ac. For CL-terms, this means that $M \rhd_1 N \rhd_1 P$ yields that M weakly reduces to P, that is, $M \rhd_{\mathrm{w}} P$. Zero intermediate steps (i.e., $n = 0$) are allowed too, which means that any one-step reduction is a weak reduction: $M \rhd_{\mathrm{w}} N$ if $M \rhd_1 N$.

The reflexive transitive closure of R is usually denoted by R^*. Thus $\rhd_{\mathrm{w}}$ is defined as $\rhd_1^*$, but the usual notation is $\rhd_{\mathrm{w}}$.

The idea that every function can be viewed as unary is not reflected by weak reduction, and that is why this relation is called "weak." Neither the axioms for combinators nor one-step reduction gives any hint as to how to compute with a term in which a combinator is followed by fewer terms than its arity. For instance, we have no clue how to apply the axiom for S to a term of the form $\mathsf{S}MN$. We will return to this issue later on and introduce other reductions. Weak reduction is the most naturally arising reduction in CL, and $\rhd_{\mathrm{w}}$ takes a central place in the theory.

In arithmetic, there is no need to worry about the order in which the arithmetic operations are calculated — as long as the groupings within the term are respected. For instance, $(5+7)\cdot(2+3)$ may be calculated stepwise as $12\cdot(2+3)$ or $(5+7)\cdot 5$, then the next step in both cases is $12\cdot 5$, and the final result is 60. There are many other ways to calculate the final result if one throws in applications of various identities (or "algebraic laws"), such as distributivity of multiplication over addition. Then $(5+7)\cdot 5 = (5\cdot 5)+(7\cdot 5)$, and further $25+(7\cdot 5) = 25+35$, etc. But our concern was simply the order of performing arithmetical operations that are explicit in an expression.

One might wonder whether the CL-terms exhibit similar behavior. The short answer is that they do. If a term N weakly reduces to P_1 and P_2, then there is always a term M to which P_1 and P_2 both weakly reduce. This property is called the *confluence of weak reduction*. We will return to this result, which is called the Church–Rosser theorem, as well as to its proof, in the next section. The notion of a normal form would make sense without the confluence of weak reduction. However, the Church–Rosser theorem means that CL-terms are well-behaved; hence, normal forms are even more important.

DEFINITION 1.3.5. (WEAK NORMAL FORM) A term M is in *weak normal form*, in *wnf* (or in *nf*, for short), iff M contains no redexes.

Weak reduction is reflexive, that is, $M \rhd_{\mathrm{w}} M$ is always a possible "further" step in a sequence of weak reductions. One-step reduction is obviously not reflexive, and it is useful to have a reduction — weak reduction — that is not just transitive, but reflexive too. However, the steps that are justified by reflexivity may be likened to having a "skip" step that can be inserted into chains of calculations at any point. Thus the above definition delineates a subset of terms that do not yield a term under $\rhd_1$.

Some obvious examples of terms in nf include the variables and the primitive combinators, each considered by itself. Some complex terms are in weak nf too. $x\mathsf{I}$ and SK are in nf, just as $\mathsf{B}(\mathsf{WI})(\mathsf{BWB})$ is. It is immediate that there are denumerably many terms that are in nf, and there are denumerably many combinators that are in nf.

Exercise 1.3.4. Prove — without assuming an infinite set of primitive constants — that there are infinitely many ($\aleph_0$-many) combinators that are in nf. (Hint: You may assume that at least S and K are among the primitives.)

Exercise 1.3.5. Consider the combinators I, B, S, K, C, W, M, B', J and D to be primitive. Is there a subset of these combinators that does not yield infinitely many combinators in nf?

There are terms that in one or more steps reduce to themselves. Let us note first that $\mathsf{II} \rhd_{\mathrm{w}} \mathsf{I}$, which is nearly II except that a copy of I is missing from the term. Replacing the identity combinator with the duplicator M we get the term MM, and $\mathsf{MM} \rhd_1 \mathsf{MM}$ — as we already noted above. Another example involves the combinator W in the term WWW. In general, any combinator that produces a left-associated term in which all arguments occur exactly once, except that one of the arguments is duplicated, can be used (possibly, together with I) to create at least one term that reduces to itself. Such a term is $\mathsf{WI}(\mathsf{WI})$, which produces a cycle via one-step reductions: $\mathsf{WI}(\mathsf{WI}) \rhd_1 \mathsf{I}(\mathsf{WI})(\mathsf{WI}) \rhd_1 \mathsf{WI}(\mathsf{WI})$. We have taken into account all the redexes at each step. Therefore, $\mathsf{WI}(\mathsf{WI}) \rhd_{\mathrm{w}} \mathsf{WI}(\mathsf{WI})$ and $\mathsf{WI}(\mathsf{WI}) \rhd_{\mathrm{w}} \mathsf{I}(\mathsf{WI})(\mathsf{WI})$, and there is no other term to which $\mathsf{WI}(\mathsf{WI})$ reduces. Of course, $\mathsf{I}(\mathsf{WI})(\mathsf{WI}) \rhd_{\mathrm{w}} \mathsf{I}(\mathsf{WI})(\mathsf{WI})$, which is another term built from W and I such that it reduces to itself.

Exercise* 1.3.6. Consider duplicators that yield a left-associated term with one argument being duplicated and all other arguments occurring exactly once. Prove that all

such duplicators (just as M and W above) are suitable to generate terms that reduce to themselves. (Hint: First, give a general description of the shape of the resulting term from the axiom of such combinators.)

These examples show that a term may reduce to itself via one or more one-step reductions. To emphasize again, $\rhd_1$ is not a reflexive relation. However, there is a CL-term M (e.g., when M is in the set of combinatory constants) such that $M \rhd_1 M$. If the set of combinators is combinatorially complete (or at least includes M or W, etc.), then there is a CL-term M such that $M \rhd_{\mathrm{w}} M$ via a nonempty sequence of $\rhd_1$ steps. This means that another — but equivalent — way to characterize weak normal forms would be to say that M is in *weak normal form* iff there is no term N, which may be the same as M, such that $M \rhd_1 N$.

The above examples showed that $\mathsf{MM} \rhd_1 \mathsf{MM}$, but not $\mathsf{WI}(\mathsf{WI}) \rhd_1 \mathsf{WI}(\mathsf{WI})$, though $\mathsf{WI}(\mathsf{WI}) \rhd_{\mathrm{w}} \mathsf{WI}(\mathsf{WI})$. Of course, M can be defined as WI, and now we make this concept more precise.

DEFINITION 1.3.6. Let Z be a combinator with axiom $\mathsf{Z}x_1 \ldots x_n \rhd M$. The CL-term Z', which is built from (primitive) combinators, *defines* Z when $\mathsf{Z}'x_1 \ldots x_n \rhd_{\mathrm{w}} M$.

This definition involves $\rhd$ and $\rhd_{\mathrm{w}}$, which suggests that there may be a difference in one-step reduction sequences when we take a combinator as undefined or when we build it up (i.e., define it) from other primitives.

Exercise 1.3.7. (a) Define B and C from S and K (only). (b) Define S from B, W, C and I.

Exercise* 1.3.8. Show that J is definable from I, B, W and C, and the latter three are definable from the first two (i.e., J and I).

It is obvious that the combinator K is *not definable* from the combinators I, B, S, C, W, M, B', J and Y. This suggests that it is important to make clear which combinators we assume to be available for us.

Exercise* 1.3.9. Prove informally (or by structural induction) that K is not definable from the other combinators mentioned in the previous paragraph.

A set of combinators determines a set of combinators that are either in the set or definable from it, which makes the next definition useful.

DEFINITION 1.3.7. Let $\mathfrak{B}$ be a finite nonempty set of combinatory constants. $\mathfrak{B}$ is called a *basis*.

The set of CL-terms, given a basis, is defined as in definition 1.1.1, where the constants are those in the basis. Exercise 1.3.7 asked you to show that B and C are definable by CL-terms over the basis $\{\mathsf{S},\mathsf{K}\}$, whereas exercise 1.3.8 asked you to show the *equivalence* of the combinatory bases $\{\mathsf{I},\mathsf{J}\}$ and $\{\mathsf{I},\mathsf{B},\mathsf{C},\mathsf{W}\}$. Just as expected, two combinatory bases are equivalent iff all the combinators definable from one are definable from the other. Defining the primitive combinators of the

other basis (in both directions) is obviously sufficient to prove the equivalence of two bases. Clearly, not all bases are equivalent, because K is not definable from $\{\mathsf{I},\mathsf{J}\}$, for instance.

The set of all possible combinators is infinite. To substantiate this claim we could simply consider all the identity combinators I^n (for all $n \in \mathbb{N}$) — there are $\aleph_0$-many of them. An interesting question to ask is whether there is a basis, preferably containing some "natural" or "not-too-complicated" combinators, that allows us to build up all the possible combinators there are.

DEFINITION 1.3.8. A combinatory basis is *combinatorially complete* iff for any function f such that $fx_1 \ldots x_n \rhd_{\mathrm{w}} M$, where M is a CL-term built from some of the variables $x_1, \ldots, x_n$, there is a combinator Z in the basis or in the set of combinators definable from the basis such that $\mathsf{Z}x_1 \ldots x_n \rhd_{\mathrm{w}} M$.

f may be thought of as if it were a proper combinator itself. However, combinatorial completeness could be defined more broadly to include certain types of improper combinators too. For example, occurrences of any proper combinator in M may be allowed. In fact, a fixed point combinator is definable from any basis that is combinatorially complete in the sense of the above definition. In fact, in the definition, M could be allowed to contain occurrences of f itself. The main purpose of the limitation to proper combinators in the definition is to exclude from our consideration all the weird improper combinators that introduce new variables into their reduct (i.e., into the term that results from an application of the combinator to sufficiently many variables). We will see in chapter 2 that a fixed point combinator is definable from S and K, which will underpin more firmly our thinking about combinatorial completeness as having a wider class of functions definable. (Cf. example 2.3.4, as well as the exercises 2.3.2 and 2.3.5.)

LEMMA 1.3.9. *The combinatory bases* $\{\mathsf{S},\mathsf{K}\}$, $\{\mathsf{I},\mathsf{B},\mathsf{C},\mathsf{W},\mathsf{K}\}$ *and* $\{\mathsf{I},\mathsf{J},\mathsf{K}\}$ *are* combinatorially complete.

Proof: First, we outline the proof that the first basis is combinatorially complete. Then we collect the facts that show that the latter two bases are equivalent to the first.
1. The idea behind showing that S and K suffice is to define a pseudo-λ-abstraction. This is usually called *bracket abstraction* or λ^*-abstraction.[9] There are many ways to define a λ^* operation, and in chapter 4, we will look at some of those. The way the λ^* is defined determines its features; hence, it affects the properties of the transition from the λ-calculus to CL. At this point our goal is merely to show that any function, in the sense of definition 1.3.8, can be simulated by a combinator over the $\{\mathsf{S},\mathsf{K}\}$ basis.

We introduce for the purpose of talking about functions such as f the notation $[x].M$. The $[x]$ is a meta-language λx, and the [ ]'s around a variable motivate the

[9] Schönfinkel's work preceded the formulation of the notion of an algorithm in a precise sense. Nevertheless, the steps in the elimination of bound variables described in his paper may be seen to amount to a bracket abstraction algorithm.

name "the bracket abstraction." M may or may not contain an occurrence of x, and $[x].M$ is interpreted as the function that returns M_x^N when applied to N. (M_x^N is M itself if $x \notin \mathrm{fv}(M)$.[10])

If f is an n-ary function, and so $f(x_1,\ldots,x_n) = M$, where M is a term over the set of variables $\{x_1,\ldots,x_n\}$, then to find a suitable combinator we iteratively form $[x_1]\ldots[x_n].f(x_1,\ldots,x_n)$, which is just f, and on the other side of the equation we form $[x_1]\ldots[x_n].M$. We apply the algorithm below starting with x_n and M. The shape of the resulting combinatory term depends on the concrete M, hence, as a general notation, we denote the result by $[x_n].M$. If $n > 1$, then the next variable is x_{n-1} and the term to which the algorithm is applied is $[x_n].M$, etc. (Because of the structure of the algorithm together with the finiteness of CL-terms, the recursion is well-founded.)

(1) $[x_i].M'$ is SKK, if M' is x_i;

(2) $[x_i].M'$ is $\mathsf{K}M'$, if x_i does not occur in M' (i.e., $x_i \notin \mathrm{fv}(M')$);

(3) $[x_i].M'$ is $\mathsf{S}([x_i].N)([x_i].P)$, if M' is NP.

We think of clauses (1)–(3) as numbered and ordered. If there are two clauses that are both applicable, then the earlier one is applied. Incidentally, (1) and (2) are never both applicable to a CL-term, and the same is true of (1) and (3). However, (3) could be applied to SKK, for example, giving $\mathsf{S}([x_i].\mathsf{SK})([x_i].\mathsf{K})$. If we do not want to rely on the order of the clauses, then $x_i \in \mathrm{fv}(NP)$ may be added as a proviso to (3).

After each of $x_1,\ldots,x_n$ has been abstracted, we have $[x_1]\ldots[x_n].M$, which is the combinator for f. To show that this is really the term that can be taken for f, we show by induction that each subterm behaves as desired.

If the subterm is x_i, and this is the variable that is being abstracted, then after an application to the term N we have to get N, and $\mathsf{SKK}N \rhd_{\mathrm{w}} N$. The other base case is when M' does not contain x_i. Then the function should return M' itself for any N, and $\mathsf{K}M'N \rhd_{\mathrm{w}} M'$. If x_i occurs in N or P when the term is (NP), then by the hypothesis of the induction, we assume that $[x_i].N$ and $[x_i].P$ are the CL-terms corresponding to N's and P's x_i abstracts. Then by applying $\mathsf{S}([x_i].N)([x_i].P)$ to Q, we get $([x_i].NQ)([x_i].PQ)$, as desired.

2. First, we note that the latter two bases are equivalent due to the results in exercise 1.3.8. Supplementing exercise 1.3.7 with two definitions, namely, SKK for I and $\mathsf{S}(\mathsf{CI})$ for W, the equivalence of the first two bases follows. S and K can define any f, and the combinators in the two other bases suffice too. $\mathfrak{qed}$

Definition 1.3.6 made precise the notion of defining a combinator from other combinators. The above lemma then implies that any *proper combinator* is definable

[10] We are cheating here a bit, because we have not yet defined substitution for λ-abstraction-like expressions. However, $[x].M$ eventually will turn out to be a combinator, so this small gap is harmless. — We use the usual notation fv to denote the set of free variables of a term. See definition A.1.12 in the Appendix.

from the basis $\{\mathsf{S},\mathsf{K}\}$. The combinatorial completeness of this basis and clear informal interpretation of S and K explain why these are the best-known combinators.

You have surely discovered that B is definable as $\mathsf{S}(\mathsf{KS})\mathsf{K}$ (which is a solution to a question in exercise 1.3.7). Now we look at the above procedure to find a definition of B, where we think of this combinator as the function f. (We take x, y and z to be x_1, x_2 and x_3, that is, $\mathsf{B}x_1x_2x_3 \rhd x_1(x_2x_3)$, but we continue to write x,y,z for the sake of brevity.)

Example 1.3.10. The resulting term is $x(yz)$, and in the first round, we want to find the CL-term for $[z].x(yz)$. $z \in \mathrm{fv}(x(yz))$, so (2) is not applicable, but neither is (1), because the term is not simply z. Then by (3), we get $\mathsf{S}([z].x)([z].yz)$. However, we are not finished yet, because there are two $[z]$'s left. $[z].x$ turns into $\mathsf{K}x$, by (2), and $[z].yz$ yields $\mathsf{S}([z].y)([z].z)$. Having put the pieces together, we obtain $\mathsf{S}(\mathsf{K}x)(\mathsf{S}([z].y)([z].z))$. Now we have gotten two $[z]$'s again, but, of course, now they are prefixed to different subterms than before. The finiteness of the subterms terminates the abstraction of $[z]$, since $[z].y$ gives $\mathsf{K}y$ and $[z].z$ gives SKK, by (2) and (1), respectively. Having gathered the bits together, we can see that the meta-term $[z].x(yz)$ refers to the CL-term $\mathsf{S}(\mathsf{K}x)(\mathsf{S}(\mathsf{K}y)(\mathsf{SKK}))$.

Before we turn to the next abstraction, it may be helpful to emphasize that $\mathsf{S}(\mathsf{K}x)(\mathsf{S}(\mathsf{K}y)(\mathsf{SKK}))$ contains *no occurrences* of z or of $[z]$. Of course, $[z]$ is a metalanguage λ, and so $[z].x(yz)$ contains only bound occurrences of z — both in $[z]$ and in the term $x(yz)$. Another name for bound variables is *apparent* variables (as distinguished from *real* variables). The CL-term $\mathsf{S}(\mathsf{K}x)(\mathsf{S}(\mathsf{K}y)(\mathsf{SKK}))$ makes the "nonreal" character of z transparent.

The next round is to find $[y][z].x(yz)$, where $[z].x(yz)$ is the pure CL-term that we already have: $\mathsf{S}(\mathsf{K}x)(\mathsf{S}(\mathsf{K}y)(\mathsf{SKK}))$. That is, we want to find the CL-term that the meta-term $[y].\mathsf{S}(\mathsf{K}x)(\mathsf{S}(\mathsf{K}y)(\mathsf{SKK}))$ denotes. Now we list — without detailed justifications — the meta-terms that successively result, and finally the CL-term itself.

(1) $\mathsf{S}([y].\mathsf{S}(\mathsf{K}x))([y].\mathsf{S}(\mathsf{K}y)(\mathsf{SKK}))$

(2) $\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}x)))(\mathsf{S}([y].\mathsf{S}(\mathsf{K}y))([y].\mathsf{SKK}))$

(3) $\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}x)))(\mathsf{S}(\mathsf{S}([y].\mathsf{S})([y].\mathsf{K}y))(\mathsf{K}(\mathsf{SKK})))$

(4) $\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}x)))(\mathsf{S}(\mathsf{S}(\mathsf{KS})(\mathsf{S}([y].\mathsf{K})([y].y)))(\mathsf{K}(\mathsf{SKK})))$

(5) $\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}x)))(\mathsf{S}(\mathsf{S}(\mathsf{KS})(\mathsf{S}(\mathsf{KK})(\mathsf{SKK})))(\mathsf{K}(\mathsf{SKK})))$

The CL-term in (5) is $[y][z].x(yz)$. The next round is to find $[x][y][z].x(yz)$, where $[y][z].x(yz)$ is the just-mentioned CL-term.

Exercise 1.3.10. Finish the example, that is, find the CL-term denoted by the meta-term $[x].\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}x)))(\mathsf{S}(\mathsf{S}(\mathsf{KS})(\mathsf{S}(\mathsf{KK})(\mathsf{SKK})))(\mathsf{K}(\mathsf{SKK})))$. (Hint: You should finally get a CL-term with 14 occurrences of S and 17 occurrences of K.)

It is already clear from the example, but even more from the exercise, that the term that results by the algorithm in the proof of lemma 1.3.9 is not $\mathsf{S}(\mathsf{KS})\mathsf{K}$. We could

obtain that term along the lines of the algorithm, if we would take some *shortcuts*, for instance, simply replacing $[z].yz$ by y. Recall that CL-terms are interpreted as functions. $z \notin \mathrm{fv}(y)$, therefore, y as a function followed by an argument is yN, and $[z].yz$ followed by an argument is $([z].yz)N$, which we stipulated (by a meta-language β-reduction) to be $[z/N]yz$, which is yN. Having applied the above algorithm, we would get the CL-term $\mathsf{S}(\mathsf{K}y)(\mathsf{SKK})$. Once this term is applied to N, we have a redex; thus, $\mathsf{S}(\mathsf{K}y)(\mathsf{SKK})N \rhd_{\mathrm{w}} \mathsf{K}yN(\mathsf{SKK}N)$. The two new redexes yield y and N, respectively. In sum, $\mathsf{S}(\mathsf{K}y)(\mathsf{SKK})N \rhd_{\mathrm{w}} yN$ too.

With this insight, we want to find again $[x][y][z].x(yz)$. First, we get $[x][y].\mathsf{S}([z].x)([z].yz)$; then we apply (1) and our shortcut to get $[x][y].\mathsf{S}(\mathsf{K}x)y$. The next step is another shortcut step, leading to $[x].\mathsf{S}(\mathsf{K}x)$. Then we get $\mathsf{S}([x].\mathsf{S})([x].\mathsf{K}x)$, and by (2) and a shortcut, we get $\mathsf{S}(\mathsf{KS})\mathsf{K}$.

The step that we labeled "shortcut" here will be called η in the λ-calculus in chapter 4. η may be thought of as an *extensionality* step, because we do not distinguish between $[x].Mx$ and M itself, when $x \notin \mathrm{fv}(M)$.

An informal rendering of the same abstraction process, which incorporates some heuristics about taking into account the effect of the combinators S and K, goes like this. We try to abstract z, and we know that $\mathsf{S}MNz \rhd_{\mathrm{w}} Mz(Nz)$. We could prefix S to the term $x(yz)$ if x would be followed by z too, as in $xz(yz)$. But to insert a dummy variable, we can use K, because $\mathsf{K}Mz \rhd_{\mathrm{w}} M$. That is, $\mathsf{K}xz(yz) \rhd_{\mathrm{w}} x(yz)$, and $\mathsf{S}(\mathsf{K}x)yz \rhd_{\mathrm{w}} \mathsf{K}xz(yz)$. $\mathsf{S}(\mathsf{K}x)yz$ is a left-associated term, and z is in the last argument place, so as the next move we just drop it. But the same is true for $\mathsf{S}(\mathsf{K}x)y$ with respect to y — we drop that variable too.

$\mathsf{S}(\mathsf{K}x)$ is very similar to $x(yz)$, if we look at x as S, y as K and z as x. So we might take instead $\mathsf{KS}x(\mathsf{K}x)$ (since $\mathsf{KS}x(\mathsf{K}x) \rhd_{\mathrm{w}} \mathsf{S}(\mathsf{K}x)$), and then $\mathsf{S}(\mathsf{KS})\mathsf{K}x$, and finally we drop x.

The example and the exercise show that the algorithm is not guaranteed to produce the shortest possible definition for a function, but it *surely produces one*. In fact, the above algorithm typically yields quite a lengthy term. Shorter terms may be obtained by including η, as well as by expanding the combinatory basis and replacing (3) with several other more refined clauses. (See chapter 4 for details.)

In arithmetic and in elementary mathematics, there is a weaker relation (than computing the value of a function) that is commonly used, namely, *equality*. For an illustration, let us consider $2^5 \cdot 3^5$. Performing the exponentiations first, the expression computes to $32 \cdot 243$, which further yields $7{,}776$. However, instead of $2^5 \cdot 3^5 \mapsto 7{,}776$ (which is, probably, not even a standard or widely used notation), simply $2^5 \cdot 3^5 = 7{,}776$ is written. Sometimes, for instance, for encryption protocols to work, it is interesting that given a positive integer — such as $7{,}776$ — that number can be factored into primes.[11] As the numerical example suggests, we can consider the *converse* of weak reduction, possibly, combined with $\rhd_{\mathrm{w}}$ itself.

[11] There is a certain similarity between the lemma 1.3.9 and the prime factorization theorem in the sense that both yield an expression that computes to the given term or number. A dissimilarity is that there are only two primitive combinators (S and K), whereas there are infinitely many primes.

DEFINITION 1.3.11. (ONE-STEP EXPANSION) If $M \rhd_1 N$, then $[N/M]P$, that is, the replacement of an occurrence of N in P by M is a *one-step expansion* of P. We will use the notation $P\,{}_1\!\lhd [N/M]P$ for one-step expansion.

Example 1.3.12. The CL-term xx can be one-step expanded to any of the following terms. $\mathsf{I}xx$, $x(\mathsf{I}x)$, $\mathsf{I}(xx)$, $\mathsf{M}x$, $\mathsf{K}xyx$, $\mathsf{K}(xx)y$, $\mathsf{K}xxx$, etc.

It might seem that expansion is completely arbitrary, but of course, it is not. The apparent arbitrariness is the result of the disappearance of the head of the reduced redex. $\rhd_1$ leaves few clues about the redex that is gone; hence, the converse step allows much freedom in creating a redex.

DEFINITION 1.3.13. (WEAK EXPANSION) The reflexive and transitive closure of one-step expansion is the *weak expansion* relation on the set of CL-terms, which is denoted by ${}_{\mathrm{w}}\!\lhd$.

Exercise* 1.3.11. Prove that weak expansion is the converse of weak reduction. That is, as we have suggested by the choice of notation, $M \rhd_{\mathrm{w}} N$ iff $N\,{}_{\mathrm{w}}\!\lhd M$.

Equality on numbers and other equality-like relations are *equivalence relations*, that is, they are *reflexive*, *transitive* and *symmetric*. The relation $\rhd_{\mathrm{w}}$ is, obviously, not symmetric; neither is weak expansion symmetric.

DEFINITION 1.3.14. (WEAK EQUALITY) The transitive reflexive symmetric closure of $\rhd_1$, the one-step reduction relation is *weak equality*, which is denoted by $=_{\mathrm{w}}$.

Weak equality may be inductively characterized by (1)–(4).

(1) If $M \rhd_1 N$, then $M =_{\mathrm{w}} N$;

(2) if M is a CL-term, then $M =_{\mathrm{w}} M$;

(3) if $M =_{\mathrm{w}} N$, then $N =_{\mathrm{w}} M$;

(4) if $M =_{\mathrm{w}} N$ and $N =_{\mathrm{w}} P$, then $M =_{\mathrm{w}} P$.

The last clause may be put succinctly as $=_{\mathrm{w}}^{+} \subseteq =_{\mathrm{w}}$, where $^{+}$ denotes the transitive closure of a binary relation.

Exercise 1.3.12. Verify that $=_{\mathrm{w}}$ is the transitive symmetric closure of $\rhd_{\mathrm{w}}$. (Hint: $=_{\mathrm{w}}$ above is characterized starting with $\rhd_1$ in (1), and so is $\rhd_{\mathrm{w}}$ in definition 1.3.4.)

Weak reduction is a stronger relation than weak equality in the sense that there are ordered pairs of terms that belong to $=_{\mathrm{w}}$ but not to $\rhd_{\mathrm{w}}$. For example, $\langle x, \mathsf{K}xy\rangle$ is in the $=_{\mathrm{w}}$ relation (i.e., $x =_{\mathrm{w}} \mathsf{K}xy$), but not in the $\rhd_{\mathrm{w}}$ relation (i.e., $x \not\rhd_{\mathrm{w}} \mathsf{K}xy$). The relationship between the binary relations we have so far on the set of CL-terms is

$$\rhd_1 \subsetneq \rhd_{\mathrm{w}} \subsetneq =_{\mathrm{w}}.$$

We have shown that both inclusions are proper (as the symbol $\subsetneq$ indicates), but we might wonder now whether we have too many pairs included in $=_{\mathrm{w}}$. Perhaps,

all CL-terms are weakly equal to each other; that is, $=_{\mathrm{w}}$ is the total relation on the set of CL-terms. In fact, not all terms are weakly equal, and this is what *consistency* means for CL. We will prove this and other consistency theorems in chapter 2.

Not only is the weak equality relation not the total relation, but we can keep adding pairs of terms to it. We briefly mentioned η, which we justified by $\mathsf{S}(\mathsf{K}M)(\mathsf{SKK})N$ weakly reducing to MN. In effect, the shortcut allowed us to identify M and $\mathsf{S}(\mathsf{K}M)(\mathsf{SKK})$ by an appeal to what happens when these terms are applied to a term N. In a similar fashion, definition 1.3.6 says that a combinator Z is defined by a (compound) combinator Z' if the application of Z' to sufficiently many arguments yields the same term as the application of Z (to the same arguments) does. As yet another example, we may consider SKK and SKS. The two terms neither are the same nor weakly reduce to each other (or to any other term, for that matter) — they are in wnf. However, $\mathsf{SKK}x \rhd_{\mathrm{w}} x$ and $\mathsf{SKS}x \rhd_{\mathrm{w}} x$. (It is not difficult to see that any other combinator would do in the place of the second occurrence of S, without affecting the result of weak reduction.)

DEFINITION 1.3.15. (EXTENSIONAL WEAK EQUALITY, 1) $[M_1/N_1,\ldots,M_m/N_m]R$ and R are *extensionally weakly equal*, denoted by $[M_1/N_1,\ldots,M_m/N_m]R =_{\mathrm{w}\zeta} R$, iff for each pair of terms M_i and N_i (where $1 \leq i \leq m \in \mathbb{N}$), there is an n_i (where $n_i \in \mathbb{N}$) such that (1) holds.

(1) For all terms $P_{i,1},\ldots,P_{i,n_i}$, there is a Q_i such that $M_iP_{i,1}\ldots P_{i,n_i} \rhd_{\mathrm{w}} Q_i$ and $N_iP_{i,1}\ldots P_{i,n_i} \rhd_{\mathrm{w}} Q_i$.

Extensionality means that the shapes of the CL-terms, or the exact reduction steps are disregarded — as long as the end result remains the same. The name for this equality is a bit lengthy, and perhaps even cumbersome; thus, we will tend to use $=_{\mathrm{w}\zeta}$ instead. The reason to use yet another Greek letter is that, roughly speaking, η is an example of extensionality, but not a sufficiently general one. In chapter 5, we add the rule ext to $EQ_{\mathfrak{B}_1}$. (This rule is sometimes labeled by ζ instead of ext.) If we would add a rule based on η to the same calculus, then a weaker system would result.

It may be useful to note that the definition stipulates that M and N reduce to the same term whenever they are applied to the same terms $P_1,\ldots,P_n$. We already saw in the discussion of associators that, if a combinator is n-ary, then forming the term with $x_1,\ldots,x_n,\ldots,x_{n+m}$ as arguments gives a term by weak reduction in which $x_{n+1},\ldots,x_{n+m}$ remain unaffected. Thus, $\mathsf{B}P_1$ does not contain a redex headed by B. However, for $n=3$, $\mathsf{B}P_1P_2P_3 \rhd_{\mathrm{w}} P_1(P_2P_3)$ and so does $\mathsf{S}(\mathsf{KS})\mathsf{K}P_1P_2P_3$. Adding further terms, $P_4,P_5,\ldots$, does not prevent the two terms from reducing to the same term. That is, the existence of some n is sufficient for the existence of *infinitely many* (larger) n's, for which the defining condition in definition 1.3.15 holds.

The P's in the definition are *arbitrary*. But in the illustrations, we looked only at what happens when two terms are applied to x or $x_1,\ldots,x_n$. However, our choice of these variables represented another sort of generality: none of them occurred in the terms like SKS or $\mathsf{S}(\mathsf{KS})\mathsf{K}$ (the definitions for I and B). In fact, the following definition yields the same concept.