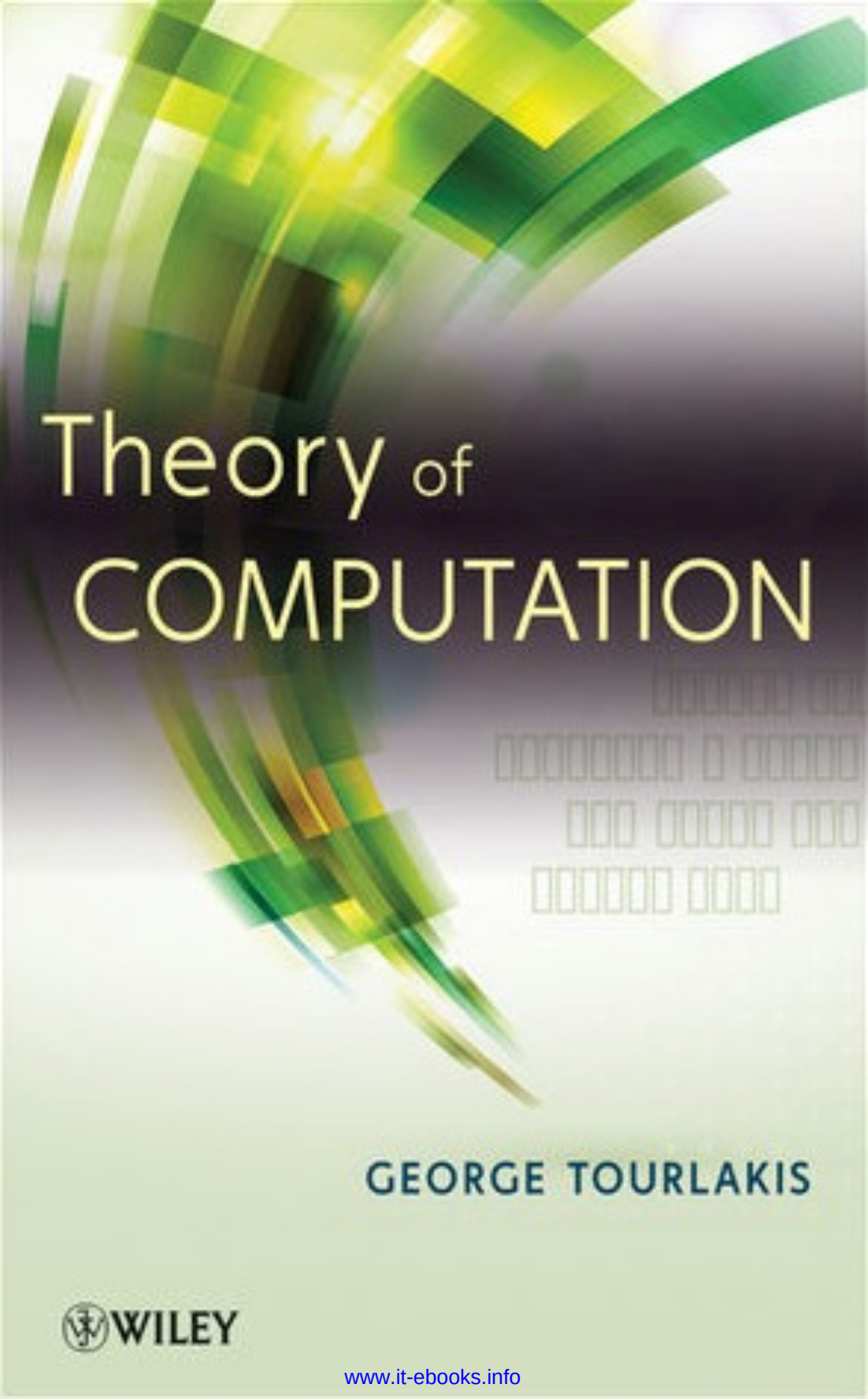


Theory of Computation

George Tourlakis
York University
Toronto, Canada

A JOHN WILEY & SONS, INC., PUBLICATION

Preface

At the intuitive level, any practicing mathematician or computer scientist —indeed any student of these two fields of study— will have no difficulty at all to recognize a *computation* or an *algorithm*, as soon as they see one, the latter defining, *in a finite manner*, computations for any given input. It is also an expectation that students of computer science (and, increasingly nowadays, of mathematics) will acquire the skill to devise algorithms (normally expressed as computer programs) that solve a variety of problems.

But how does one tackle the questions "is there an algorithm that solves such and such a problem for all possible inputs?" —a question with a potentially "no" answer— and also "is there an algorithm that solves such and such a problem via computations that take no more *steps* than some (fixed) polynomial function of the input length?" —this, too, being a question with a, potentially, "no" answer.

Typical (and tangible, indeed "interesting" and practically important) examples that fit the above questions, respectively, are

- "is there an algorithm which can determine whether or not a given computer program (the latter written in, say, the C-language) is *correct*?"[1]

[1] A "correct" program produces, *for every input*, precisely the output that is expected by an *a priori* specification.

and

- "is there an algorithm that will determine whether or not any given Boolean formula is a tautology, doing so via computations that take no more *steps* than some (fixed) polynomial function of the input length?"

For the first question we have a definitive "no" answer,[2] while for the second one we simply do not know, at the present state of knowledge and understanding of what "computing" means.[3]

But what do we mean when we say that "there is *no algorithm* that solves a given problem" —with or without restrictions on the algorithm's computation lengths? This appears to be a much harder statement to validate than "there *is* an algorithm that solves such and such a problem" —for the latter, all we have to do is *to produce such an algorithm* and a proof that it works as claimed. By contrast, the former statement implies a, mathematically speaking, *provably failed search* over the entire set of *all algorithms*, while we were looking for one that solves our problem.

One evidently needs a precise definition of the concept of algorithm that is neither experiential, nor technology-dependent in order to assert that we encountered such a failed "search". This directly calls for a *mathematical theory* whose objects of study include *algorithms* (and, correspondingly, *computations*) in order to construct such sets of (all) algorithms within the theory and be able to reason about the membership problem of such sets. This theory we call the *theory of computation*. It contains tools which, *in principle*, can "search"[4] the set of all algorithms to see whether a problem is solvable by one; or, more ambitiously, to see if it can be solved by an algorithm whose computations are "efficient" —under some suitable definition of efficiency.

The theory of computation is the *metatheory* of computing. In the field of computing one computes: that is, develops programs and large scale software that are well-

[2]There is some interesting "small print" here! As long as the concept of *algorithm* is identified with that of, say, the Shepherdson-Sturgis "machines" of this volume —or for that matter with Turing machines— then the answer is definitely a "no": There is a simple mathematical proof that we will see later on, that no Shepherdson-Sturgis machine (nor a Turing machine) exists that solves the problem. Now, such an identification has been advocated by Alonzo Church as part of his famous belief known as "Church's Thesis". If one accepts this identification, then the result about the non-existence of a Shepherdson-Sturgis machine that solves the problem is tantamount to the non-existence of an *algorithm* that does so. However, Church's "thesis" is empirical, rather than provable, and is not without detractors; cf. Kalmár (1957). Suffice it to say that *this* statement is mathematically valid: *No program, written in any programming language, which is equivalent in expressive power to that of our Shepherdson-Sturgis machines, exists that solves the problem.*

[3]There is substantial evidence that the answer, if discovered, will likely be "no".

[4]The quotes are necessary since it is not precisely a *search* that one performs. For example, the unsolvability —by any algorithm— of the *program correctness problem* is based on a so-called *reduction* technique that we will learn in this volume. A reduction basically establishes that a problem A is solvable by algorithmic means if we *assume* that we have a "black-box" algorithmic solution —that we may "call" just as we call a built-in function— of another problem, B. We say that "A is reduced (or reducible) to B". If we now *know* (say via a previous mathematical proof of the fact) that A cannot be algorithmically solved, then nor can B! We will, as a starting point, show the unsolvability by algorithmic means, certainly not by any Shepherdson-Sturgis machine, of a certain "prototype" problem, known as *the halting problem*, "$x \in K$?". This will be done by a technique akin to Cantor's *diagonalization*. After this, many reduction arguments are effected by showing that K is reducible to a problem A. This renders A unsolvable!

documented, correct, efficient, reliable and easily maintainable. In the (meta)theory of computing one tackles the fundamental questions of the *limitations of computing*, limitations that are intrinsic rather than technology-dependent.[5] These limitations may rule out outright the existence of algorithmic solutions for some problems, while for others they rule out efficient solutions.

Our approach is anchored on the concrete (and assumed) practical knowledge about general computer programming attained by the reader in a first year programming course, as well as the knowledge of discrete mathematics at the same level. The next natural step then is to develop the metatheory of *general computing*, building on the computing experience that we have assumed the reader attained. This will be our chapter on computability, that is, the most general *meta*theory of computing. We develop this metatheory via the programming formalism known as Shepherdson-Sturgis *Unbounded Register Machines* (URM) —which is a straightforward abstraction of modern high level programming languages. Within that chapter we will also explore a restriction of the URM programming language, that of the *loop programs* of A. Meyer and D. Ritchie. We will learn that while these loop programs can only compute a very small subset of "all the computable functions", nevertheless are *significantly more than adequate* for programming solutions of any "practical", computationally solvable, problem. For example, even restricting the nesting of loop instructions to *as low as two*, we can compute —in principle— enormously large functions, which with input x can produce outputs such as

$$\left.\begin{matrix} 2^{2^{\cdot^{\cdot^{\cdot^{x}}}}} \end{matrix}\right\} 10^{350000} \text{ 2's} \tag{1}$$

The qualification above, "in principle", stems from the enormity of the output displayed in (1) —even for the input $x = 0$— that renders the above function way beyond "practical".

The chapter —after spending considerable care in developing the technique of *reductions*— concludes by demonstrating the intimate connection between the *unsolvability phenomenon* of computing on one hand, and the *unprovability phenomenon* of proving within first-order logic (cf. Gödel (1931)) on the other, when the latter is called upon to reason about "rich" theories such as (Peano's) arithmetic —that is, the theory of natural numbers, equipped with: the standard operations (plus, times); relations (less than); as well as with the principle of mathematical induction.

What to include and what not to include in an introductory book on the theory of computation is a challenge that, to some extend, is resolved by the preferences of the author. But I should like to think that the choices of topics made in this volume are more rational than simply being manifestations of "preference".

The overarching goal is to develop for the reader a "first-order" grounding in the fundamentals, that is, the theoretical limitations of computing in its various models of computation, from the most general model —the URM— down to the finite automaton.

[5]However this metatheory is called by most people "theory". Hence the title of this volume.

We view the technique of *reductions* as fundamental in the analysis of limitations of computing, and we spend a good deal of space on this topic, a variant of which (polynomial-time reductions) the student of computer science will encounter in Subsection 5.1.2 and will re-encounter in later studies as well, for example, in a course on algorithms and complexity. On the other hand, we do not hesitate to omit combinatorial topics such as "Post's correspondence problem", which only leads to specialized results (e.g., the algorithmic unsolvability of detecting ambiguity in context free languages) that we feel embody a less fundamental technical interest. Our emphasis is on laying the foundational tools and concepts that allow us to carry out a mathematical analysis of, and acquire a thorough understanding of, theoretical limitations of computing in both their absolute manifestation (uncomputability) and also in their relative manifestation (complexity and "intractability").

Consistent with our stated goal and emphasis, we purposely give short shrift to the area of so-called "positive" results, apart from a few familiarization examples of "programming" with URMs, loop programs, FA, NFA, and PDA. This is not a course about writing algorithms, but mostly about what algorithms *cannot do at all* and about what *they have a lot of trouble doing*. For example, results of Chapter 5 immediately imply that, in general, FORTRAN-like programs that allow nesting of the loop instruction equal to just *three* have highly *impractical* run times; certainly as high as[6]

$$2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \Big\} x \text{ 2's}$$

Thus, we leave out "applications" such as lexical scanners via finite automata; automata-minimization; parsing of context free languages using LL, LR, recursive-descend, and other parsers; and defer them to a later course on *compiler writing tools* —these topics do not belong here. We would rather concentrate on what is *foundationally* important and omit what is not.

Another challenge is where to *start* building this metatheory. What should be our abstraction of a computer program? It should be a straightforward observation that since this metatheory, or "theory" as we nickname it, *abstracts* computing practices —in order to analyze and study said abstractions mathematically— the student must have encountered in the first instance *the concrete counterparts* of these *abstractions* for the latter to make any sense.

It is hardly the case that, prior to the second year of study, students have "programmed" scanners or parsers. Rather, students have programmed solutions for less specialized problems, using a *high level general purpose* language such as C/C++, Java, possibly Pascal, etc. They never programmed an automaton, a push-down automaton, or anything like a Turing machine (unless they have taken up machine language in the first year).

Yet the overwhelming majority of the literature develops the "theory of computation", in a manner of speaking, backwards —invariably starting with the theory of

[6]See 5.2.0.47 and 5.2.0.49. L_3 programs have run times bounded by Ackermann's $A_3^k(x)$, for some $k > 0$.

finite automata, as if automata is precisely what the reader was programming in his[7] first university course on programming. We apply this principle: Before the student studies the (meta)theory, he must have attained a good grasp of the *practice* that this theory attempts to dissect and discuss. Thus, it is natural to start our story with the (meta)theory of *general purpose computer programs*.

Because of these considerations, our first chapter is on URMs and computability. The choice of URMs as an abstraction of general-purpose computing —a relative latecomer (cf. Shepherdson and Sturgis (1963)) in the search for a good answer to "what would be a good technology-independent model of computation?"— also connects well with the experience of the student who will come to this course to learn what makes things tick in programming, and why some things do not tick at all. He most likely learned his programming via a high level language like C or Java rather than through *machine language*. The ubiquitous Turing machine (Turing (1936, 1937)) is more like machine language, indeed, is rather even less user-friendly.[8] It offers no advantage at this level of exposition, and rather presents an obscure and hard-to-use (and hard to "arithmetize"[9]) model of computation that one *need not* use as the *basis* of computability. On the other hand it lends itself well to certain studies in complexity theory and is an eminently usable tool in the proof of Cook's theorem (cf. Subsection 5.1.3). So we will not totally avoid the Turing machine!

We turn to the formulaic topics of a book on *Automata and Languages* — Chapter 3— only after we become familiar, to some extent, with the (general) computability theory, including the special computability theory of more "practical" functions, the primitive recursive functions. *Automata* are introduced as a very restricted *programming formalism*, and their limitations (in expressivity) and their associated languages are studied.

It is often said, with justification, that a course in theory of computation has as side-effect the firming up of the student's grasp of (discrete) mathematical techniques and mathematical reasoning, as well as the ability to apply such techniques in computer science and beyond. Of course, it cannot be emphasized enough that the student of a theory of computation course must be equipped already with the knowledge expected to be acquired by the successful completion of a one-semester course on discrete mathematics. This required background knowledge is often encapsulated, retold, and aspects of it are emphasized, in the space of a few pages at the front-end of a book like this. This is the ubiquitous "Chapter 0" of many books on the subject. In the case of the present book I would like, most of all, to retell two stories, *logic* and *induction*, that I often found being insufficiently developed in the student's "toolbox", notwithstanding earlier courses he may have taken. Thus, in Subsection 1.1.1 we develop the notational and how-to parts of elementary predicate logic in the space of some 20 pages, paying special attention to correctness of exposition. Section 1.4 presents the induction principle on the natural numbers in two steps: One, how

[7]Pronouns such as "he", "his", "him" are, *by definition*, gender-neutral in this volume and are used solely for textual convenience.

[8]Machine language can manipulate *numbers*, whereas a Turing machine can only manipulate *digits*!

[9]This verb will make sense later.

to use its various forms, and a proof of their equivalence to the least (positive) integer principle. Two, we argue, at the intuitive level, why induction *must* be a *valid principle* after all![10] We also go over concepts about sets and related notation, as well as relations and functions, very quickly since they do not need much retelling. We will also introduce quickly and in an elementary fashion a topic likely not encountered by the reader in the typical "discrete math" course: the distinction between two infinities —*countable* and *uncountable*— so that we can have an excuse to introduce the reader to Cantor's ingenious (and simple) *diagonalization* argument, that recurs in one or another shape and form, over and over, in the computability and complexity part of the theory of computation.

On intuitive arguments; "formalization" and why a course in theory cannot be taught exclusively *by hand-waving*: The main reason that compels us to teach (meta)theory in a computer science curriculum is not so much to prevent the innocent from trying to program a solution for the halting problem (cf. 2.5.0.16), just as we do not teach courses in geometry just to prevent circle-squaring "research". Rather, formal mathematical methods used in a course in the theory of computation, more so than the results themselves, are transferable skills that the student becomes endowed with, which equip him to model and mathematically analyze concrete phenomena that occur in computation, and through a mathematical process of reasoning to be able to recognize, understand, and correlate such phenomena. These formal methods, skills and results, put the "science" keyword into *computer science*.

Intuition, obtained through experience, is invaluable, of course, and we often argue intuitively *before* we offer a proof of a fact. But: one *cannot* have "proof-by-intuition".

We have included in this volume a good amount of *complexity theory* that will likely be mostly skipped whenever the book is called upon to serve a second year course on the theory of computation. There are a few "high level complexity" results already in Section 2.7 using diagonalization (cf. 2.7.1.9 and 2.7.1.11). Later, quite a bit is developed in Chapter 5, including the concept of $\mathscr{NP}$-completeness and Cook's theorem; an account of Cobham's class of *feasibly computable functions* (mostly delegated to the Exercises section, 5.3); and some elements of the hierarchy theory of the primitive recursive functions culminating in the rather startling fact that we cannot algorithmically solve the *correctness problem* of FORTRAN-like programs even if we restrict the *nesting of loops* to just *two levels*. FORTRAN-like languages have as abstract counterpart the *loop programs* of Meyer and Ritchie (1967) that we study in the chapters on computability (2nd) and complexity (5th).

Were I to use this book in a second year course in the theory of computation I would skim quickly over the mathematical "prerequisites" chapter, and then cover 2.1–2.7, parts of 2.10, certainly Gödel's incompleteness theorem and its relation to *uncomputability*: 2.11 —but not 2.11.1. I would then cover only as much as time permits from Chapter 3 on finite automata; certainly the pumping lemma, consistent

[10]In so doing I will be sure to let the student know that I am not squaring the circle: Induction is not a provable principle of the arithmetic of Peano, it is an axiom. However, this will not stop us from arguing its *plausibility*, i.e., why it is a *reasonable*, "natural" axiom.

with my view that this is a "course" about what *cannot* be done, or cannot be done "easily", rather than a toolbox for how to *do* things. The latter is deferred to a course and book on algorithms.

In a more advanced course where one can proceed faster, I would want also to cover the sections on creative sets and the recursion theorem, and also as much complexity theory as possible from Chapter 5, starting with the material leading to Cook's theorem.

The reader will forgive the many footnotes, which some will assess as bad style! There is always a story within a story, the "... and another thing ...", that is best delegated to footnotes.

The style of exposition that I prefer is informal and conversational and is expected to serve well not only the readers who who have the guidance of an instructor, but also those readers who wish to learn the elements of the theory of computation on their own. I use several devices to promote understanding, such as frequent "pauses" that anticipate questions and encourage the reader to rethink an issue that might be misunderstood if read but not studied and reflected upon. Additionally, I have included numerous remarks, examples and embedded exercises (the latter in addition to the end-of-chapter exercises) that reflect on a preceding definition or theorem. All pauses are delimited by "**Pause.**" and ◄

The stylized "winding road ahead" warning,  , that I first saw in Bourbaki's books (Bourbaki (1966)) and have used in my other books, delimits a passage that is too *important* to skim over.

On the other hand, I am using  to delimit passages that I could not resist including, but, frankly, can be skipped (unless you are curious).

There are over 200 end-of-chapter exercises and 41 embedded ones. Many have hints and thus I refrained from (subjectively) flagging them for level of difficulty. After all, as one of my mentors, Alan Borodin, used to say to us (when I was a graduate student at the University of Toronto), "attempt all exercises; but definitely do the ones you cannot do".

GEORGE TOURLAKIS

Toronto
November 2011

CHAPTER 1

MATHEMATICAL FOUNDATIONS

In this chapter we will briefly review tools, methods and *notation* from mathematics and logic, which we will directly apply throughout the remaining of this volume.

1.1 SETS AND LOGIC; NAÏVELY

The most elementary elements from "set theory" and logic are a good starting point for our review. The quotes are necessary since the term *set theory* as it is understood today applies to the *axiomatic* version, which is a vast field of knowledge, methods, tools and research [cf. Shoenfield (1967); Tourlakis (2003b)]—and this is not what we outline here. Rather, we present the standard notation and the elementary operations on sets, on one hand, and take a brief look at infinity and the *diagonal method* of Cantor's, on the other. Diagonalization is a tool of significant importance in computability. The tiny fragment of concepts from set theory that you will find in this section (and then see them applied throughout this volume) are framed within Cantor's original "naïve set theory", good expositions of which (but far exceeding our needs) can be found in Halmos (1960) and Kamke (1950).

We will be forced to interweave our exposition of concepts from set theory with concepts—and notation—from elementary logic, since all mathematics is based on

Theory of Computation. By George Tourlakis

logical deductions, and the vast majority of the literature, from the most elementary to the most advanced, employs logical notation; e.g., symbols such as "$\forall$" and "$\exists$".

The term "set" is *not defined*,[11] in either the modern or in the naïve Cantorian version of the theory. Expositions of the latter, however, often ask the reader to think of a *set* as just a synonym for the words "class",[12] "collection", or "aggregate". Intuitively, a set is a "container" along with its contents—its *elements* or *members*. Taken together, contents and container, are viewed as a *single mathematical object*. In mathematics one deals only with sets that contain mathematical objects (so we are not interested in sets of mice or fish).

Since a set is itself an *object*, a set may contain sets as elements.

All the reasoning that one does in order to develop set theory—even that of the naïve variety—or any part of mathematics, including all our reasoning in this book, utilizes mathematical logic. Logic is the mathematics of reasoning and its "objects" of study are predominantly mathematical "statements" or "assertions"—technically known as *formulae*[13]—and mathematical proofs. Logic can be applied to mathematics either experientially and informally—learned via practice as it were—or formally. The predominance of mathematical writings apply logic informally as a vehicle toward reaching their objectives.[14] Examples of writings where logic is formally applied to mathematics are the volumes that Bourbaki wrote, starting here [Bourbaki (1966)]. More recent examples at the undergraduate and graduate levels are Gries and Schneider (1994) and Tourlakis (2003b) respectively.

In this volume we apply logic informally. An overview is provided in the next subsection.

1.1.1 A Detour via Logic

As is customary in mathematics, we utilize *letters*, upper or lower case, usually from near the end of the alphabet (u, v, y, x, z, S, T, V) to *denote*, that is, to *name* mathematical objects—in particular, *sets*.

By abuse of language we say that u, v, y, x, z, S, T, V *are* (rather than *denote* or *name*) objects. These letters function just like the variables in algebra do; they are *object-variables*.

[11] The reader who has taken Euclidean geometry in high school will be familiar with this parallel: The terms "point", "line", and "plane" are not defined either, but we get to know them intimately through their properties that we develop through mathematical proofs, starting from Euclid's axioms.

[12] In axiomatic set theory a "class" is a kind of collection that may be so "large" that it technically fails to be a set. The axioms force sets to be "small" classes.

[13] More accurately, a "statement" and a formula are two different things. However, the latter mathematically "encodes" the former.

[14] Despite the dangers this entails, as Gödel's incompleteness theorems exposed [Gödel (1931)], modern mathematicians are confident that their subject and tools have matured enough, to the point that one can safely apply logic, once again, post-Gödel, informally. For example, Kunen states in his article on set-theoretic *combinatorics*, Kunen (1978), "A knowledge of [formal] logic is neither necessary, nor even desirable".

As is the case in algebra, the variables x, y, z are not the only objects set theory studies. It also studies numbers such as $0, 1, -7$ and π, matrices such as $\left(\begin{smallmatrix} 0 & 1 \\ 1 & 2 \end{smallmatrix}\right)$ and objects that are the results of function applications such as 7^{23000}, x^{y^z} and 2^x.

Unlike axiomatic set theory, which introduces its objects via formal constructions, naïve set theory allows us to use, "off the shelf", all the mathematical objects such as the above, as well as, of course, objects that are sets such as $\{2, 3, \{1\}\}$ and $A \cup B$.[15]

Logicians like to call mathematical objects *terms*. We utilize in this book the generic names t and s (with primes or subscripts, whenever we need more than two such names) to refer to arbitrary terms that we do not want to be specific about.

1.1.1.1 Definition. The *simplest* possible relations of set theory are of just *two forms*: $t \in s$—read "t is a member of s" or "t belongs to s"—and $t = s$, read "t is equal to s", where, as we indicated above, t and s are any terms whatsoever.

These relations are the *atomic formulae* (of set theory). The qualifier "atomic" refers to two facts:

- These two types cannot be expressed (simulated) in terms of simpler relations by using the notation and tools of logic.
- Using these two relations as building blocks we can *construct* every possible formula of set theory as we will explain shortly. □

1.1.1.2 Example. $x \in y$, $u = v$, $z \in S$ and $3 \in z$ and $2^z = y^3$ are atomic formulae.

$\mathbb{N}$, the set of all *natural numbers* (i.e., all the numbers that we obtain by starting at 0 and repeatedly adding 1: 0, 1, 2, 3, 4, ...), is an important constant in naïve set theory.

By "$\mathbb{N}$ ... *is* an important constant" we mean, of course, via the habitual abuse of language exercised by mathematicians, the accurate "$\mathbb{N}$ ... *denotes* (or *names*) an important constant".

Here is an example that uses $\mathbb{N}$ in an atomic formula: $-7 \in \mathbb{N}$. Incidentally, this formula makes (i.e., encodes) a false statement; we say the *formula is false*.

One may form this basic formula as well, $\mathbb{N} = \bigcup_{i=0}^{\infty}\{i\}$, where the meaning of the symbols "$\{\ldots\}$" and "$\bigcup_{i=0}^{\infty}$" will be introduced later in this section.

Yet another example is $\{1\} \in \{2, 1\}$—a false statement (formula) as we will be able to determine soon. □

Logic (and mathematics) contain much more complex formulae than those of the atomic variety. The added complexity is achieved by repeatedly "gluing" atomic formulae together employing as glue the logical, or *Boolean, connectives*

$$\neg, \wedge, \vee, \rightarrow, \equiv$$

[15] Notation for objects such as $\{\ldots\}$ and $x \cup y$ will be reviewed shortly.

and the *quantifiers*

$$\forall, \exists$$

As we have noted already, unlike the case of naïve set theory—where we take for granted the *a priori* presence of *all* objects of mathematics, such as 3, -7, $\mathbb{N}$ and $x^{y^{z^u}}$ — axiomatic set theory needs no *a priori* existence of any objects. Starting just with the relations $x \in y$ and $x = y$ it uses powerful rules, which can be used to build not only all formulae of set theory, but also all the objects of mathematics that we are familiar with, such as the above-mentioned and many others.

What about arithmetic? The arithmetical objects of "pure" (Peano) arithmetic are the variables, constants, and outputs of functions applied on objects that we have already built. What are its formulae? If we are thinking of pure arithmetic, which is studied outside set theory, then we may choose as atomic formulae all those that can be built from the three start-up relations $z = x + y$, $z = x \times y$ and $z = x^y$: new atomic formulae result by substituting arbitrary (arithmetical) objects for variables. Note that the equality relation is obtained from $z = x + y$ by substituting 0 for y.

All formulae of arithmetic can be built, starting from the atomic ones, as explained in the general Definition 1.1.1.3 below. This assertion is revisited in Subsection 2.11.1.

Gödel showed in Gödel (1931) that the atomic formula $z = x^y$ is, well, *not atomic*: It can be simulated (built) within pure arithmetic starting just with $z = x + y$ and $z = x \times y$.

The "practicing mathematician" prefers to work within an "impure" arithmetic, where he has access to sets and their notations, operations, and properties. In particular, this impure arithmetic employs set variables and, more generally, set objects in addition to number variables and number objects.

Throughout this volume a formula (whether specific to set theory or to any other area in mathematics, such as arithmetic—pure or impure) will be *denoted* by an upper case calligraphic letter, such as $\mathscr{A}, \mathscr{B}, \mathscr{F}, \mathscr{G}$.

We now indicate how formulae are put together using brackets, connectives, and quantifiers, employing atomic formulae as basic building blocks. The definition below is generic, thus unified: it applies to the structure of all formulae of mathematics. The choice of atomic formulae (which presupposes an *a priori* choice of mathematical symbols, such as $0, +, \in$) and of types of variables is what determines whether we build set theory formulae, pure or impure arithmetic formulae, or "other".

1.1.1.3 Definition. A set theory formula is one of:

(1) An atomic formula (1.1.1.1).

(2) $(\neg\mathscr{A})$, where $\mathscr{A}$ is known to be[16] a formula.

[16]I.e., to stand for one. Thus, the expression "$(\neg\mathscr{A})$" is constructed by writing "(", followed by writing "$\neg$", followed by *writing in full* whatever $\mathscr{A}$ names, and finally writing ")".

(3) $(\mathscr{A} \wedge \mathscr{B})$, where $\mathscr{A}$ and $\mathscr{B}$ are known to be formulae.

(4) $(\mathscr{A} \vee \mathscr{B})$, where $\mathscr{A}$ and $\mathscr{B}$ are known to be formulae.

(5) $(\mathscr{A} \to \mathscr{B})$, where $\mathscr{A}$ and $\mathscr{B}$ are known to be formulae.

(6) $(\mathscr{A} \equiv \mathscr{B})$, where $\mathscr{A}$ and $\mathscr{B}$ are known to be formulae.

(7) $((\forall x)\mathscr{A})$, where $\mathscr{A}$ is known to be a formula and x is any variable.

(8) $((\exists x)\mathscr{A})$, where $\mathscr{A}$ is known to be a formula and x is any variable. We say in the last two cases that "$\mathscr{A}$ is the *scope* of Qx, where Q is $\forall$ or $\exists$".

We call $\forall$ the *universal* and $\exists$ the *existential* quantifiers. We will extend the terminology "quantifier" to apply to the compound symbols $(\forall x)$ or $(\exists)$. □

1.1.1.4 Definition. (Immediate Predecessors) Let $\mathscr{F}$ be a formula. By 1.1.1.3 it has one of the forms (1)–(8). If it is of type (1), then it has no *immediate predecessors*—i.e., it was not built using connectives or quantifiers from simper formulae. If it has the forms (2)–(8), then in each case its immediate predecessors are the formulae $\mathscr{A}$ and $\mathscr{B}$ [the latter enters in cases (3)–(6)] that were used to build it. We use the acronym *ip* for *immediate predecessors*. □

The presence of brackets guarantees that the decomposition or deconstruction of a formula into its immediate predecessors is unique. This fact can be proved, but it is beyond our aims so we will not do so here [see Bourbaki (1966); Enderton (1972); Tourlakis (2008, 2003a)]. Logicians refer to it as the *unique readability* of a formula.

1.1.1.5 Example. Here are some formulae:

$x \in y$, $3 = z$, $z = x^w$—by (1),

$(\neg x = y)$—by (1), followed by an application of (2); we usually write this more simply as "$x \neq y$",

$(x \in y \vee z = x^w)$—by (1), followed by an application of (4),

$((\forall x)z = x^w)$—by (1), followed by an application of (7),

$(x = 0 \to x = 0)$—by (1), followed by an application of (5), and

$(x = 0 \to ((\forall x)x = 0))$—by (1), followed by an application of (7) to obtain $((\forall x)x = 0)$, and then by an application of (5).

The reader should check that we inserted brackets precisely as prescribed by Definition 1.1.1.3. □

1.1.1.6 Remark. (Building a formula) If $\mathscr{F}$ is (stands for, that is) a formula we can deconstruct it according to Definition 1.1.1.3 using a natural process.

Initialize: Write down $\mathscr{F}$. Flag it *pending*.

Repeat this process *until it cannot be carried further*:

{

Write down, *above* whatever you have written so far, the *ip* of all *pending* formulae (if they have ip); and *remove* the flag "pending" from the latter. *Add* the flag to the ones you have just written.

}

The process is terminating since we write *shorter and shorter formulae* at every step (*and* remove the flags); we cannot do this forever!

Clearly, if we now review *from top to bottom* the sequence that we wrote, we realize that it traces forward the process of constructing $\mathscr{F}$ by repeated application of Definition 1.1.1.3. This top-down view of our "deconstruction" is a *formula-construction sequence* for $\mathscr{F}$.

For example, applying the process to the last formula of the preceding example we get:

$x = 0$
$x = 0$
$((\forall x)x = 0)$
$(x = 0 \to ((\forall x)x = 0))$

where one copy of $x = 0$ was contributed by the bottom formula and the other (at the top) by $((\forall x)x = 0)$.

Going forward we can discard copies that we do not need. Thus a valid formula construction is also this one:

$x = 0$
$((\forall x)x = 0)$
$(x = 0 \to ((\forall x)x = 0))$

Indeed, we validate the first formula in the sequence via (1) of 1.1.1.3; the second using the first and (7); and the last one using the first two and (5). □

A term such as x^2 has x as its only input variable. An atomic formula such as $z \in \mathbb{N}$ has z as its only input variable, while the (atomic) formula $x + y = y^w$ has x, y and w as input variables. Whenever we want to draw attention to the input variables—say, x, u, S and z—of a term t or a formula $\mathscr{A}$ we will write $t(x, u, S, z)$ or $\mathscr{A}(x, u, S, z)$, respectively. This is entirely analogous to writing "$f(x, z) = x^2 + \sin z$" in order to name the expression (term) $x^2 + \sin z$ as a function $f(x, z)$ of the two listed variables.

1.1.1.7 Definition. (Input Variables—in Terms) All the variables that occur in a term—other than an x that occurs in a term of the form $\{x : \ldots\}$ (which is a set object that will be introduced shortly)—are input variables. □

1.1.1.8 Example. Thus, the term x has x as its only input variable; while the term 3 has no input variables. $x^{2^{z^y}}$ has x, z, y as its input variables. We will soon introduce terms (set objects) such as $\{x : x = 0\}$. This object, which the reader may recognize as a fancy way to simply write $\{0\}$, has no input variables. □

1.1.1.9 Definition. (Input Variables—in Formulae) A variable occurrence[17] in an atomic formula $t \in s$ or $t = s$ is an *input occurrence* precisely if it is an input occurrence in one of the terms t and s. Thus, "$0 \in \{x : x = 0\}$" has no input variables while "$x = 0$" has one.

Formation rules (2)–(6) in Definition 1.1.1.3 "*preserve*" the input occurrences of variables in the constituent formulae $\mathscr{A}$ and $\mathscr{B}$ that we join together using one of $\neg, \wedge, \vee, \rightarrow, \equiv$ as glue. On the other hand, each quantifier $(\forall z)$ or $(\exists z)$ *forces* each occurrence of a variable as described below to become non-input:

- The occurrence z in the quantifier
- Any occurrence of z in the scope of said $(\forall z)$ or $(\exists z)$

Thus, if we start with $\mathscr{A}(x, y, z)$, of inputs x, y, z, the new formula $((Qy)\mathscr{A}(x, y, z))$, where Q stands here for one of $\forall, \exists$, has only x and z as input variables. □

We have carefully referred to *occurrences*, rather than *variables*, in the above definition. A *variable can* be both input and non-input. An *occurrence cannot* be both. For example, in $(x = 0 \rightarrow (\forall x)x = 0)$ the first x-occurrence is input; the last two are non-input. The *variable* x is both.

Thus "x is an input/non-input variable" (of a formula) means that there are occurrences of x that are input/non-input.

The standard name utilized in the literature for input variables is *free variables*. Non-input variable occurrences are technically called *bound occurrences*, but are also called *apparent occurrences*, since even though they are visible, they are not allowed—indeed it makes no sense—to receive arguments (input). This is analogous to the "Σ-notation" for sums: $\sum_{i=1}^{3} i$ means $1 + 2 + 3$. While we can "see" the variable i, it is not really there![18] It cannot accept inputs. For example, "$\sum_{2=1}^{3} 2$" is total nonsense.

The jargon input/non-input *is deliberately chosen: We may substitute terms* only *in those variable occurrences that are free (input).*

If $\mathscr{F}$ is some formula and $x, y, z, \ldots$ is the *complete* list of variables that occur in it, we can draw attention to this fact by writing $\mathscr{F}(x, y, z, \ldots)$. If $x, y, z, \ldots$ is a list of variables such that *some*[19] among them occur in $\mathscr{F}$, then we indicate this by $\mathscr{F}[x, y, z, \ldots]$.

In the context of $\mathscr{F}[x, y, z, \ldots]$ [or $\mathscr{F}(x, y, z, \ldots)$], $\mathscr{F}[t_1, t_2, t_3, \ldots]$ [correspondingly $\mathscr{F}(t_1, t_2, t_3, \ldots)$] stands for the formula obtained from $\mathscr{F}$ by replacing each *original* occurrence of $x, y, z, \ldots$ in $\mathscr{F}$ by the terms $t_1, t_2, t_3, \ldots$ respectively.

Some people call this operation *simultaneous* or *parallel* substitution. Thus, if $\mathscr{F}[x, y]$ names "$x = y$", whereas t_1 is $y+1$, and t_2 is 5, then $\mathscr{F}[t_1, t_2]$ is "$y+1 = 5$" and *not* "$5+1 = 5$". The latter result would have been obtained if we first substituted

[17] For example, in $x = x$ the variable x has two occurrences.

[18] A fact demonstrated strongly by the explicit form of the sum, $1 + 2 + 3$.

[19] "Some" includes "none" and "all" as special cases.

t_1 in x to obtain $y + 1 = y$, and *then* substituted t_2 in y to obtain $5 + 1 = 5$. If we are to do this "simultaneous substitution" right, then we must not substitute t_2 into the y to the left of "="; this y is *not* "original".

Observe also that if x does *not* occur in $\mathscr{F}[x]$, then $\mathscr{F}[t]$ is just the *original* $\mathscr{F}$.

Before we turn to the *meaning* (and naming) of the connectives and quantifiers, let us agree that we can get away with much fewer brackets than Definition 1.1.1.3 prescribes. The procedure to do so is to *agree* on connective and quantifier "priorities" so that we know, in the absence of brackets, which of the two connectives/quantifiers is supposed to "win" if they both compete to apply on the same part in a formula.

By analogy, a high school student learns the convention that "$\times$ has a higher priority than $+$", thus $2 + 3 \times 4$ means $2 + (3 \times 4)$—that is, $\times$ rather than $+$ claims the part "3".

Our convention is this: The connective $\neg$ as well as the quantifiers $\forall$ and $\exists$ have the highest priority, equal among the three. In order of decreasing priority, the remaining *binary* connectives[20] are listed as $\wedge, \vee, \rightarrow, \equiv$. If two binary connectives compete to glue with a subformula, then the higher-priority one wins. For example, assuming that $\mathscr{A}$ has already in place all the brackets that are prescribed by Definition 1.1.1.3, then $\ldots \rightarrow \mathscr{A} \vee \cdots$ means $\ldots \rightarrow \Big(\mathscr{A} \vee \cdots$, while $\ldots \neg\mathscr{A} \wedge \cdots$ means $\ldots \Big(\neg\mathscr{A}\Big) \wedge \cdots$.

If *two instances of the same binary connective* compete to glue with a subformula, then *the one to the right* wins. For example, assuming that $\mathscr{A}$ has all the brackets prescribed by Definition 1.1.1.3 in place, then $\ldots \rightarrow \mathscr{A} \rightarrow \cdots$ means $\ldots \rightarrow \Big(\mathscr{A} \rightarrow \cdots$.

Similarly, if any of $\neg, \forall, \exists$ compete for a part of a formula, again *the one to the right* wins. E.g., $\ldots \neg(\forall x)(\exists y)\mathscr{A} \cdots$ means $\ldots \Big(\neg\Big((\forall x)\big((\exists y)\mathscr{A}\big)\Big)\Big) \cdots$, where once again we assumed that $\mathscr{A}$ has all the brackets prescribed by Definition 1.1.1.3 already in place.

How do we "compute" the truth or falsehood of a formula? To begin with, to succeed in this we must realize that just as a function gives, in general, different outputs for different inputs, in the same way the "output" of a formula, its *truth-value*, can only be computed, in general, if we "freeze" the input variables. For each such frozen instance of the input side, we can compute the output side: true or false.

But where do the inputs come from? For areas of study like calculus or arithmetic the answers are easy: From the set of real numbers—denoted by $\mathbb{R}$—and the set of natural numbers respectively.

For set theory it sounds easy too: From the set of all sets!

If it were not for the unfortunate fact that "the set of all sets" does not exist, or, to put it differently, it is a *non-set class* due to its enormity, we could have left it

[20] "Binary" since they each glue *two* subformulae.

at that. To avoid paradoxes such as "a set that is *not* a set"—cf. Section 1.3 on diagonalization for an insight into why some collections cannot be sets—we will want to take our inputs from a (comfortably large) *set* in any given set-theoretic discussion: the so-called *reference set* or *domain*.

1.1.1.10 Remark. The mathematician's intuitive understanding of the statement "$\mathscr{F}$ is *true* (resp. *false*)" is that "$\mathscr{F}$ is true (resp. false) for *all* the possible values of the free (input) variables of $\mathscr{F}$".

Thus, if we are working in arithmetic, "$2n+1$ is odd" means the same thing as "it is true that, for all $n \in \mathbb{N}$, $2n+1$ is odd". "$2^n > n$" again omits an implied prefix "it is true that, for all $n \in \mathbb{N}$". An example of a false statement *with* input variables is "$2n$ is odd". □

1.1.1.11 Definition. An *instance* of a formula $\mathscr{F}$, in symbols $\mathscr{F}'$, is a formula obtained from $\mathscr{F}$ by replacing *each* of its variables by some value from the relevant reference set.

Clearly, $\mathscr{F}'$ is variable-free—a so-called *closed formula* or *sentence*—and therefore it has a well-defined truth-value: exactly one of *true* or *false*.

Sometimes we use more explicit notation: An instance of $\mathscr{G}(x, y, z, \ldots)$ or of $\mathscr{G}[x, y, z, \ldots]$ is $\mathscr{G}(i, j, k, \ldots)$ or $\mathscr{G}[i, j, k, \ldots]$, respectively, where $i, j, k, \ldots$ are objects (constants) from the reference set.

$\mathscr{F}'$ and $\mathscr{G}'$ are *consistent* or *common* instances of $\mathscr{F}$ and $\mathscr{G}$ if *every* free variable that appears in both of the latter receives the same value in both instances. □

1.1.1.12 Example. Let $\mathscr{A}$ stand for "$x(x+1)$ is even", $\mathscr{B}$ stand for "$2x+1$ is even" and $\mathscr{C}$ stand for "x is even", where x is a variable over $\mathbb{N}$. Then,

$\mathscr{A}$ is true,

$\mathscr{B}$ is false, and

$\mathscr{C}$ is neither true, nor false.

The lesson from this is that if the truth-value of a formula depends on variables, then *not true* is *not* necessarily the same as *false*. □

We will not be concerned with how the truth-value of atomic formulae is "computed"; one can think of them as entities analogous to "built-in functions" of computer programming: *Somehow, the way to compute their true/false output is a matter that a hidden procedure (alternatively, our math knowledge and sophistication) can do for us.*

Our purpose here rather is to describe how the *connectives and quantifiers* behave toward determining the truth-value of a complex formula.

In view of Remark 1.1.1.10, the road toward the semantics of $\mathscr{A} \vee \mathscr{B}$, $((\forall x)\mathscr{A})$, etc., passes through the semantics of arbitrary instances of these; namely, we need to only define the meaning of $\mathscr{A}' \vee \mathscr{B}'$, $((\forall x)\mathscr{A})'$, etc., respectively.

1.1.1.13 Definition. (Computing with Connectives and Quantifiers) Let $\mathscr{A}$ and $\mathscr{B}$ be any formulae, and $\mathscr{A}'$ and $\mathscr{B}'$ be arbitrary *common* instances (1.1.1.11).

(1) $\neg\mathscr{A}'$—pronounced "not $\mathscr{A}'$"—is true *iff*[21] $\mathscr{A}'$ is false.

(2) $\mathscr{A}' \vee \mathscr{B}'$—pronounced "$\mathscr{A}'$ or $\mathscr{B}'$"—is true iff either $\mathscr{A}'$ is true or $\mathscr{B}'$ is true, or *both* (so-called *inclusive or*).

(3) $\mathscr{A}' \wedge \mathscr{B}'$—pronounced "$\mathscr{A}'$ and $\mathscr{B}'$"—is true iff $\mathscr{A}'$ is true *and* $\mathscr{B}'$ is true.

(4) $\mathscr{A}' \to \mathscr{B}'$—pronounced "if $\mathscr{A}'$, then $\mathscr{B}'$"—is true iff either $\mathscr{A}'$ is false *or* $\mathscr{B}'$ is true, or *both*.[22]

(5) $\mathscr{A}' \equiv \mathscr{B}'$—pronounced "$\mathscr{A}'$ iff $\mathscr{B}'$"—is true just in case[23] $\mathscr{A}'$ and $\mathscr{B}'$ are both true or both false.

(6) The instance $(\forall x)\mathscr{A}(i_1, \ldots, i_m, x, j_1, \ldots, j_n)$—which is pronounced "for all x, $\mathscr{A}(i_1, \ldots, i_m, x, j_1, \ldots, j_n)$ (holds)"[24]— is true iff, for *all* possible values k of x from the domain, $\mathscr{A}(i_1, \ldots, i_m, k, j_1, \ldots, j_n)$ is true.

(7) The instance $(\exists x)\mathscr{A}(i_1, \ldots, i_m, x, j_1, \ldots, j_n)$ —which is pronounced "for some x, $\mathscr{A}(i_1, \ldots, i_m, x, j_1, \ldots, j_n)$ (holds)"—is true iff, for *some* value k of x from the domain, $\mathscr{A}(i_1, \ldots, i_m, k, j_1, \ldots, j_n)$ is true. □

1.1.1.14 Remark. (Truth Tables) The content of the preceding definition—cases (1)–(5)—is normally captured more visually in table form (we have removed the primes for readability):

$\mathscr{A}$	$\mathscr{B}$	$\neg\ \mathscr{A}$	$\mathscr{A} \vee \mathscr{B}$	$\mathscr{A} \wedge \mathscr{B}$	$\mathscr{A} \to \mathscr{B}$	$\mathscr{A} \equiv \mathscr{A}$
f	**f**	**t**	**f**	**f**	**t**	**t**
f	**t**	**t**	**t**	**f**	**t**	**f**
t	**f**	**f**	**t**	**f**	**f**	**f**
t	**t**	**f**	**t**	**t**	**t**	**t**

We read the table as follows: First, the symbols **t** and **f** stand for the values "true" and "false" respectively. Second, the two columns to the left of the vertical line || give all possible pairs of values (outputs) of $\mathscr{A}$ and $\mathscr{B}$. Third, below $\neg\mathscr{A}$, $\mathscr{A} \vee \mathscr{B}$, etc., we list the computed truth-values (of the formulae of the first row) that correspond to the assumed $\mathscr{A}$ and $\mathscr{B}$ values.

The odd alignment under $\neg\mathscr{A}$ is consistent with all the others: It emphasizes the placement of the "result" under the "operator"—here $\neg$—that causes it. □

[21] *if and only if*

[22] Other approaches to "implication" are possible. For example, the *Intuitionists* have a different understanding for $\to$ than that of the majority of mathematicians, who adopt the classical definition above.

[23] A synonym of "iff".

[24] The verb "holds" means "is true".

1.1.1.15 Remark. According to Remark 1.1.1.10,

$$(\forall x)\mathscr{A}(y_1,\ldots,y_m,x,z_1,\ldots,z_n) \text{ is true} \qquad (\dagger)$$

means precisely this:

For every choice of the i_l and j_r from the reference set,
$(\forall x)\mathscr{A}(i_1,\ldots,i_m,x,j_1,\ldots,j_n)$ is true $\qquad (*)$

By 6 of Definition 1.1.1.13, $(*)$ means

For every choice of the i_l and j_r from the reference set,
and
for *all* possible values k of x from the domain,

$$\mathscr{A}(i_1,\ldots,i_m,k,j_1,\ldots,j_n) \text{ is true}$$

The above, and hence also $(\dagger)$, translate via Remark 1.1.1.10 as

$$\mathscr{A}(y_1,\ldots,y_m,x,z_1,\ldots,z_n) \text{ is true} \qquad (\ddagger)$$

Iterating this observation yields that $(\ddagger)$ is an equivalent statement to the one we obtain by quantifying universally any—in particular, *all*—of the variables $y_1,\ldots,y_m$, $x,z_1,\ldots,z_n$ of $\mathscr{A}$. That is,

> Adding or removing a "$(\forall x)$" at the leftmost end of the formula makes no difference to the latter's meaning.

Hm. This begs the question: Then what do we need the universal quantifier for? □

1.1.1.16 Example. We note easily that, say, with $\mathbb{R}$ (the reals) as our domain, $x = 0 \to x = 0$ is true (cf. 4 in 1.1.1.13). However, $x = 0 \to (\forall x)x = 0$ is not, since its instance $0 = 0 \to (\forall x)x = 0$ is false: to the left of $\to$ we have true, while to the right we have false.

Thus, adding or removing a "$(\forall x)$" to *parts* of a formula *can* make a difference in meaning! The universal quantifier is useful after all. □

Carrying around the predicate "is true" all the time is annoying. We will adopt immediately the mathematician's jargon: *Simply stating* "$\mathscr{A}(x,y,\ldots)$" is *synonymous* to "$\mathscr{A}(x,y,\ldots)$ is true" or "$\mathscr{A}(x,y,\ldots)$ holds".

1.1.1.17 Example. Let $\mathbb{N}$ be our domain. Then,

$$(\exists x)y < x \qquad (1)$$

is true. It says that "for every y, an x exists[25] such that $y < x$". According to 1.1.1.15 there is an *implied* $(\forall y)$ at the beginning of the formula.

[25]That is, "for every *value of* y, a *value of* x exists". The mathematician is used to the sloppy language that omits "*value of*". It is clear that he does *not* refer to the variables themselves, but rather refers to their values.

Note that there is *no* single value of x that makes (1) true (because $\mathbb{N}$ has no upper bound). For each value n of y, $n+1$ is an appropriate value of x (so are $n+2$, $2n+1$, etc.)

How can we write down the (false) statement that *one* value of x works for all y?

$$(\exists x)(\forall y) y < x$$

A final remark: How do we write that "there exists a *unique* x such that $\mathscr{A}$ is true" (where $\mathscr{A}$ is any formula)?

$$(\exists x)\Big(\mathscr{A}[x] \wedge \neg(\exists z)(\mathscr{A}[z] \wedge x \neq z)\Big)$$

Fortunately, there is a short form for the above ($\exists!$ reads "for some unique")

$$(\exists! x)\mathscr{A}$$ □

1.1.1.18 Example. The reader probably already knows that there is redundancy in the chosen set of connectives, that is, some of them can be simulated by the others.

For example, it is immediate by comparing (4), (1), and (2) in 1.1.1.13, that $\mathscr{A} \to \mathscr{B}$ is the same as (has the same meaning as) $\neg\mathscr{A} \vee \mathscr{B}$. Similarly, $\mathscr{A} \equiv \mathscr{B}$ is the same as $(\mathscr{A} \to \mathscr{B}) \wedge (\mathscr{B} \to \mathscr{A})$.

Even $\mathscr{A} \wedge \mathscr{B}$ can be expressed via $\neg$ and $\vee$ as $\neg(\neg\mathscr{A} \vee \mathscr{B})$. This is easiest to see, perhaps, via a truth-table:

$\mathscr{A}$	$\mathscr{B}$	$\neg$	($\neg$	$\mathscr{A}$	$\vee$	$\neg$	$\mathscr{B}$)	$\mathscr{A} \wedge \mathscr{B}$
f	**f**	(4)**f**	(1)**t**		(3)**t**	(2)**t**		**f**
f	**t**	(4)**f**	(1)**t**		(3)**t**	(2)**f**		**f**
t	**f**	(4)**f**	(1)**f**		(3)**t**	(2)**t**		**f**
t	**t**	(4)**t**	(1)**f**		(3)**f**	(2)**f**		**t**

The numbers such as "(1)**t**" in the first row indicate order of evaluation using the operator at the top of the column. Comparison of the column labeled (4) with the last column shows that each of $\mathscr{A} \wedge \mathscr{B}$ and $\neg(\neg\mathscr{A} \vee \mathscr{B})$ yield the same output for any given value-pair of $\mathscr{A}$ and $\mathscr{B}$. Thus we proved the equivalence of the $\mathscr{A} \wedge \mathscr{B}$ and $\neg(\neg\mathscr{A} \vee \mathscr{B})$. This result is known as "de Morgan's Law". □

1.1.1.19 Exercise. Prove that $\mathscr{A} \vee \mathscr{B}$ can be expressed as $\neg(\neg\mathscr{A} \wedge \mathscr{B})$. This is the "other" (technically *dual* of) de Morgan's Law. □

1.1.1.20 Example. We know [(7) of Definition 1.1.1.13] that $(\exists x)\mathscr{A}(z_1, \ldots, z_m, x, y_1, \ldots, y_n)$ means

$$\text{For every choice of } i_1, \ldots, i_m, j_1, \ldots, j_n, \tag{1.1}$$
$$\text{there is a } k \text{ such that} \tag{1.2}$$
$$\mathscr{A}(i_1, \ldots, i_m, k, j_1, \ldots, j_n) \tag{1.3}$$

Now, the *negation* of "there is a k such that $\mathscr{A}(i_1, \ldots, i_m, k, j_1, \ldots, j_n)$" is

$$\text{"no } k \text{ makes } \mathscr{A}(i_1, \ldots, i_m, k, j_1, \ldots, j_n) \text{ true"} \tag{1}$$

that is,

$$\text{all } k \text{ make } \neg\mathscr{A}(i_1, \ldots, i_m, k, j_1, \ldots, j_n) \text{ true} \tag{2}$$

(2) says the same thing as

$$(\forall x)\neg\mathscr{A}(i_1, \ldots, i_m, x, j_1, \ldots, j_n) \tag{3}$$

(1.2)–(1.3), (1), and (3) yield (via 1.1.1.13)

$$(\exists x)\mathscr{A}(i_1, \ldots, i_m, k, j_1, \ldots, j_n) \equiv \neg(\forall x)\neg\mathscr{A}(i_1, \ldots, i_m, x, j_1, \ldots, j_n)$$

By (1.1) and 1.1.1.10, we have

$$(\exists x)\mathscr{A}(z_1, \ldots, z_m, x, y_1, \ldots, y_n) \equiv \neg(\forall x)\neg\mathscr{A}(z_1, \ldots, z_m, x, y_1, \ldots, y_n)$$

for short

$$(\exists x)\mathscr{A} \equiv \neg(\forall x)\neg\mathscr{A} \qquad \square$$

1.1.1.21 Exercise. Prove that $(\forall x)\mathscr{A} \equiv \neg(\exists x)\neg\mathscr{A}$. $\square$

1.1.1.22 Remark. We note that $\mathscr{A} \to \mathscr{B}$ is true, in particular, when *no* instance of $\mathscr{A}$ is true, i.e., when $\mathscr{A}$ is *false*—in all its instances, that is. In this case the so-called *classical* or *material implication* holds "vacuously", *even if there is no connection between $\mathscr{A}$ and $\mathscr{B}$ at all and even if $\mathscr{B}$ is not true!* For example, if $\mathscr{A}$ is $0 \neq 0$ and $\mathscr{B}$ is "in every Euclidean triangle the sum of the angles equals 9π", then $(\mathscr{A} \to \mathscr{B})$ is true. The same holds if $\mathscr{B}$ stands for "n is even". The latter has both true and false instances over $\mathbb{N}$, but that is immaterial. In each chosen instance, $\mathscr{A}' \to \mathscr{B}'$ is true—that is (1.1.1.10), $\mathscr{A} \to \mathscr{B}$ is true.

Equally disturbing is the fact that while both sides of the arrow might be true, though *totally unrelated*, the *implication* will be true, as in $\mathscr{C} \to \mathscr{D}$ where $\mathscr{C}$ is $0 = 0$ and $\mathscr{D}$ is "in every Euclidean triangle the sum of the angles equals 2π".

The Intuitionists, a school of thought founded on the writings of Kronecker, Brouwer and Heyting, do not like this state of affairs. The *intended meaning*, or *intended semantics*, for their $\mathscr{A} \to \mathscr{B}$, connects the hypothesis $\mathscr{A}$ strongly to the conclusion $\mathscr{B}$. The meaning, informally speaking, is that, from a *proof* (intuitionistic proof, of course!) of $\mathscr{A}$, a proof for $\mathscr{B}$ can be *constructed*.

We are not going to say what *is* an intuitionistic "proof", as this is of no concern to us. As a matter of fact, "proof" (for classical logic) will be defined only later (see 1.1.1.34). At present, let the reader think of "proof" as a process used to establish that a formula holds. Nevertheless, the above stated *intentions* regarding (intuitionistic) proofs are a clear enough indication of how strongly $\mathscr{A}$ and $\mathscr{B}$ must be related before the Intuitionist will agree to write $\mathscr{A} \to \mathscr{B}$.

In particular, in intuitionistic logic $\rightarrow$ and $\vee$ do not relate as in 1.1.1.18 above. In fact, in both logics $\mathscr{A} \rightarrow \mathscr{A}$ holds, however, in intuitionistic logic $\mathscr{A} \vee \neg\mathscr{A}$ does *not* hold! Or as we say, the *law of the excluded middle* does *not* hold. Intuitively speaking, the reason behind this phenomenon is that the intuitionistic proofs are so structured that, to establish that a formula $\mathscr{A} \vee \mathscr{B}$ holds, in general you need to construct a proof of one of $\mathscr{A}$ or $\mathscr{B}$.

For example, the following classical proof that there are *irrational* numbers[26] a, b such that a^b is rational is *unacceptable intuitionistically*.

Take $a = b = \sqrt{2}$. If this works, done. If not, that means that $\sqrt{2}^{\sqrt{2}}$ is *irrational*. Well, then take $a = \sqrt{2}^{\sqrt{2}}$ and $b = \sqrt{2}$. Now $a^b = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^2 = 2$. A rational. End of proof.

Above, $\mathscr{A}$ is "$\sqrt{2}^{\sqrt{2}}$ is rational" and $\mathscr{B}$ is "$\sqrt{2}^{\sqrt{2}}$ is irrational". We used the (classical) fact that $\mathscr{A} \vee \mathscr{B}$ is true, since one or the other of $\mathscr{A}$ and $\mathscr{B}$ is (classically) true. However, classically, we do not need to know which is which!

A thorough exposition of intuitionistic logic can be found in the advanced book of Schütte ([Schü]). □

1.1.1.23 Example. Suppose that x does not occur free in $\mathscr{A}[x]$.

Pause. Ensure that this is consistent with the notation introduced in Definition 1.1.1.11.◀

Then

$$\mathscr{A}[x] \equiv (\forall x)\mathscr{A}[x] \tag{1}$$

Indeed, let $y, z, w, \ldots$ be the complete list of free variables of $\mathscr{A}$, where x is not one of them. To verify (1) we need to show that for any choice of values $k, l, m, \ldots$ from the domain

$$\mathscr{A}(k, l, m, \ldots) \equiv (\forall x)\mathscr{A}[x, k, l, m, \ldots] \tag{2}$$

that is,

$$\text{both sides of (2) are } \mathbf{t} \text{ or both are } \mathbf{f}. \tag{3}$$

We need to analyze $(\forall x)\mathscr{A}[x, k, l, m, \ldots]$. It says

"for all n in the domain, $\mathscr{A}[n, k, l, m, \ldots]$ is true"

But this is true exactly when $\mathscr{A}(k, l, m, \ldots)$ is, since n does not affect the output: the variable x is non-input in $\mathscr{A}$. □

1.1.1.24 Exercise. Suppose we drop the condition "suppose that x does not occur free in $\mathscr{A}[x]$" above. Does (1) still hold? You must provide the "why"! □

[26] That is, not rational. A rational number has the form, by definition, p/q where both p and $q \neq 0$ are integers.

1.1.1.25 Exercise. In view of 1.1.1.15, "$\mathscr{A}[x]$ is true" iff "$(\forall x)\mathscr{A}[x]$ is true". So, is this observation not all that we need to assert (1)? See also the previous exercise. □

Atomic formulae contain no Boolean connectives at all. On the other hand, formulae with a *leading* quantifier, $\forall$ or $\exists$, contain no *explicit* Boolean connectives: any Boolean connectives they may have are "hidden" in the quantifier's scope. Thus, one may view these two categories of formulae as "*atomic, but from a Boolean point of view*", meaning you cannot split them into *simpler* ones by simply *removing a Boolean connective*. For example, if you start with $(\forall x)(x = 0 \vee x = y)$ and remove the $\vee$, you get the nonsensical $(\forall x)(x = 0\ x = y)$. Logicians call these "Boolean atomic formulae" *prime* formulae but also *Boolean variables*.

1.1.1.26 Definition. (Prime Formulae; Tautologies) A *prime formula* or *Boolean variable* is either an atomic formula, or a formula with a leading quantifier.

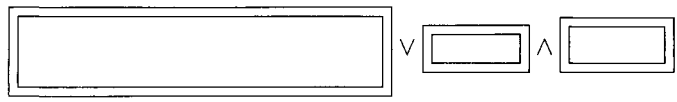
If a formula $\mathscr{F}$—when viewed as a *Boolean combination* of prime formulae, that is, as a formula built from prime formulae using only the formation rules (2)–(6) of 1.1.1.3—evaluates as **t** according to the truth table 1.1.1.14, for *all possible assumed* truth-values of its Boolean variables, then we call it a *tautology* and write this concisely as $\models_{taut} \mathscr{F}$. □

1.1.1.27 Remark. Every formula is built by appropriately applying Boolean glue on a number of prime formulae: Indeed, in any formula built according to 1.1.1.3 we can identify all its *maximal* prime subformulae—that is prime subformulae not contained in larger prime subformulae.

For example, in $(\forall x)(x = 0 \to (\exists y)x = y) \vee w = u \wedge u = 2^x$ we may indicate the prime subformulae by "boxing" them as below.

$$\boxed{(\forall x)(\boxed{x = 0} \to \boxed{(\exists y)\boxed{x = y}})} \vee \boxed{\boxed{w = u}} \wedge \boxed{\boxed{u = 2^x}} \qquad (1)$$

Double-boundary boxes enclose maximal prime formulae. The Boolean structure of (1) is



Only maximal prime formulae contribute to the Boolean structure and express the original formula as a Boolean combination of prime formulae glued together by connectives. The non-maximal prime formulae are hidden *inside maximal ones.*

The italicized claim above follows directly from an adaptation of the "deconstruction" in 1.1.1.6:

Just replace the step

> Write down, *above* whatever you have written so far, the *ip* of all *pending* formulae (if they have ip); and *remove* the flag "pending" from the latter.

by

Write down, *above* whatever you have written so far, the *ip* of all *non-prime pending* formulae (if they have ip); and *remove* the flag "pending" from the latter.

Conversely, a Boolean combination of prime formulae is a formula in the sense of 1.1.1.3: Indeed, a Boolean combination is an expression built by applying (2)–(6) in the context of a formula-construction (cf. 1.1.1.6) with starting points prime formulae, rather than atomic formulae. Since a prime formula *is* a formula, the process leads to a formula. See Exercise 1.8.1 for a rigorous proof (that you will supply, equipped with the induction tool). □

The quantifier "(for all possible) *assumed*" in Definition 1.1.1.26 is significant. It means that we *do not* compute the actual (intrinsic) truth-values of the constituent Boolean variables (in the domain that we have in mind)—*even if they do have such a value*; note that $x = y$ does not.

Rather, we *assume* for each prime formula, *all the—in principle—possible output values*; that is, *both* of **t** and **f**.

For example, for the determination of tautologyhood of a formula, where $x = x$ enters as a Boolean variable, we *assume* two possible output values, **t** *and* **f** even though we *know* that its *intrinsic* value is **t**.

In particular, $x = x$ is *not* a tautology.

Pause. Ensure that this last observation fits with 1.1.1.26!◀

We indicate this fact by writing $\not\models_{taut} x = x$.

Assuming all possible truth-values of a prime formula (rather than attempting to compute "the" value) is tantamount to allowing the *Boolean structure* and Boolean structure alone—that is how the connectives have glued the formula together—to determine the truth-value via truth tables (1.1.1.14).

1.1.1.28 Example. Some tautologies: $x = 0 \to x = 0$, $(\forall x)\mathscr{A} \vee \neg(\forall x)\mathscr{A}$, $x = y \to x = y \vee z = 2^{2^w}$.
Some non-tautologies: $x = 0 \to x = 5$, $(\forall x)x = x \vee (\forall y)y = y$, $x = y \to x = w \vee z = 2^{2^w}$. For example, looking at the value assumptions below—or value *assignments*, as is the accepted term,

$x = y := \mathbf{t}$
$x = w := \mathbf{f}$, and
$z = 2^{2^w} := \mathbf{f}$,

we see that $x = y \to x = w \vee z = 2^{2^w}$ evaluates as **f**, thus it is indeed not a tautology. Incidentally, I used ":=" to denote (truth) value assignment. □

1.1.1.29 Definition. (Tautological Implication) We say that $\mathscr{A}_1, \ldots, \mathscr{A}_n$ *tautologically imply* $\mathscr{B}$ iff $\models_{taut} \mathscr{A}_1 \to \mathscr{A}_2 \to \ldots \to \mathscr{A}_n \to \mathscr{B}$.

We write $\mathscr{A}_1, \ldots, \mathscr{A}_n \models_{taut} \mathscr{B}$ in this case. $\mathscr{B}$, we say, is (the result of) a tautological implication from $\mathscr{A}_1, \ldots, \mathscr{A}_n$. □

1.1.1.30 Remark. We note that a tautological implication *preserves truth*, from left to right. See exercise below. □

1.1.1.31 Exercise. Let $p_1, \ldots, p_m$ be all the Boolean variables that appear in the formulae $\mathscr{A}_1, \ldots, \mathscr{A}_n, \mathscr{B}$.

Show that $\mathscr{A}_1, \ldots, \mathscr{A}_n \models_{taut} \mathscr{B}$ iff every set of values assigned to the Boolean variables that makes *all* the $\mathscr{A}_i$ **t**, also makes $\mathscr{B}$ **t**. □

1.1.1.32 Example. All of the following are correct: $x = 0 \models_{taut} x = 0, x < y, y < z \models_{taut} x < y \wedge y < z$, $x = y \models_{taut} \mathscr{A} \vee \neg\mathscr{A}$ (no matter what $\mathscr{A}$ stands for). $\mathscr{A} \models_{taut} \mathscr{B} \to \mathscr{A}$ is also correct, since $\mathscr{A} \to \mathscr{B} \to \mathscr{A}$, that is, $\mathscr{A} \to (\mathscr{B} \to \mathscr{A})$ can be readily verified as a tautology using either 4 of 1.1.1.13 or the truth table (1.1.1.14). A shortcut is to just consider the two assumed values, **t** and **f**, for $\mathscr{A}$. The second makes the formula true outright, while the first makes the bracketed subformula **t**, hence the whole formula **t**.

This is incorrect: $x < y, y < z \models_{taut} x < z$ since choosing

$x < y := \mathbf{t}$

$y < z := \mathbf{t}$, and

$x < z := \mathbf{f}$

we see that $x < y \to y < z \to x < z$ evaluates as **f**, so it is not a tautology. □

1.1.1.33 Remark. (Capture of a Free Variable) Let $\mathscr{A}(x)$ stand for $(\exists y) y \neq x$—recall that $y \neq x$ is short for $\neg y = x$. It states (i.e., codifies the statement) "for any value of x there is a value of y that is different". Assuming that our domain is $\mathbb{N}$, this is clearly a true statement. So is $\mathscr{A}(z)$, obtained by substituting z for x, or, in programming jargon, "calling" $\mathscr{A}(x)$ with argument z.

What about $\mathscr{A}(y)$? This is $(\exists y) y \neq y$ which is evidently false: "there is a value of y which is different from itself"!

This is unfortunate because, intuitively, what $\mathscr{A}(x)$ says should have nothing to do with the name of the input variable!

What happened here is that when we substituted y for x, the free y was *captured*—i.e., *became bound*—by a lurking quantifier, $(\exists y)$: y got into the latter's scope.

We should never allow such substitutions since, as this example shows, they may *change the intended meaning of the resulting formula. In general, a substitution into $\mathscr{F}[x]$ that results into $\mathscr{F}[t]$ should* not *be allowed, if the term t contains a free variable y that will become bound (captured) after the substitution.*

Is there a workaround? Yes!

Consider an instance $(\exists x)\mathscr{F}(i_1, \ldots, i_m, x, j_1, \ldots, j_n)$ of

$$(\exists x)\mathscr{F}(z_1, \ldots, z_m, x, w_1, \ldots, w_n) \tag{1}$$

and a *consistent* instance (cf. 1.1.1.11) $(\exists u)\mathscr{F}(i_1, \ldots, i_m, u, j_1, \ldots, j_n)$ of

$$(\exists u)\mathscr{F}(z_1, \ldots, z_m, u, w_1, \ldots, w_n) \tag{2}$$

where u is a new variable. The two instances are equivalent, since if $x = k$ works for the first, then $u = k$ works for the second, and vice versa. The instance being arbitrary, we get the equivalence of (1) and (2), that is

$$(\exists x)\mathscr{F}(z_1, \ldots, z_m, x, w_1, \ldots, w_n) \equiv (\exists u)\mathscr{F}(z_1, \ldots, z_m, u, w_1, \ldots, w_n)$$

In view of 1.1.1.20 and 1.1.1.21, or directly using a similar argument as above, one sees at once that changing the bound variable of a universal quantifier into a brand new variable does not change the meaning either.

This leads to the so-called "*variant theorem*" of logic, that

Changing a bound variable in a formula into a new *(i.e., not already used in the formula) variable does not change the meaning: the original and the new formulae are equivalent.*

But then, given $\mathscr{A}[x]$, we can *always* effect $\mathscr{A}[t]$ with impunity as long as we *rename* out of harm's way, *before the substitution takes place*, all the original bound variables:[27] All we need to do in a successful renaming is to ensure that none of them is the same as a free variable of t. Strictly speaking, in $\mathscr{A}[t]$ we do not have the original formula $\mathscr{A}$, but a *variant* of the original—since we have renamed the latter's bound variables. Nevertheless since the old and the new are equivalent, we will use the same name for both, $\mathscr{A}$. □

We now turn to what a mathematician or computer scientist does with logic: He writes proofs.

1.1.1.34 Definition. (Proofs) A *proof* is a *construction process* that builds a *finite length sequence* of true formulae, *one at a time*. We normally write this sequence vertically on the page, with explanatory annotation. Three simple rules regarding what formula we *may* write at each (construction-) step govern the process. We may write

(1) A formula that we *know* as, or *accept* as, true.

(2) A formula that is a tautological implication of formulae already written in the course of the proof.

(3) $(\forall x)\mathscr{A}(\ldots, x, \ldots)$ provided $\mathscr{A}(\ldots, x, \ldots)$ has already been written in the course of the proof.

Any formula $\mathscr{A}$ that appears in a proof we call a *theorem*. We say that the proof "established" or *proved* the theorem $\mathscr{A}$. □

A theorem follows from certain axioms Γ. Saying just "theorem" does not indicate this dependence, unless what is the relevant set of axioms is clearly understood from the context. If in doubt, or if we are discussing theorems of various theories

[27] This is a sufficient and straightforward overkill. In principle, we only *need* to rename those bound variables that are referenced in those quantifiers that will capture a variable in t, if we do nothing.

simultaneously, then we must name the applicable axiom set in each case by saying "Γ-theorem" or "theorem from Γ".

It is clear from what we have developed so far, that steps (2) and (3) *preserve truth* (cf. 1.1.1.15). An application of step (3) is called application of *generalization*, or rule *Gen*, as we will say.

What exactly is going on in step (1)? Well, we know that some formulae are true because we recognize them as such outright, without the help of any complicated process; they are "initial" truths—or *initial theorems*—such as $x = x$, which is true in *all mathematics*, or $x + 1 \neq 0$, which is *true in a specific theory*: arithmetic of the natural numbers. These initial theorems, whether they are universal or theory-specific, are called *axioms*.

1.1.1.35 Remark. All axioms are "atomic theorems", that is, they are obtained by an application of (1)—they are not "results" of the application of (2) or (3) on previous theorems written in the course of a proof.

As indicated in the  -passage above, we have two types of axioms.

(a) Those that are true because of the way the formulae that express them are put together, using connectives and quantifiers. These axioms are *not* specific to any branch of mathematics: *They hold for all mathematics.*

We call such axioms *logical.*[28]

With some ingenuity, a very small set of formulae[29] can be chosen, among the *universally* or *absolutely true* formulae, to serve as logical axioms. Read on!

For example, $x = x$ and $(\forall x)\mathscr{A}[x] \to \mathscr{A}[t]$ are such universal truths, and we will adopt both as logical axioms.

(b) A formula $\mathscr{A}$ is a *nonlogical* axiom in a *mathematical theory* provided it is taken as an important *start-up truth of the theory*—an "atomic theorem"— *not* because of its form, but rather because of *what it says* in connection with the various symbols that are peculiar to the theory.

For example, $x + 1 \neq 0$ is an important start-up truth—a nonlogical axiom—of (Peano) arithmetic over $\mathbb{N}$. There is nothing to render it "universal"; in fact, it is not a true statement if our domain is either the reals, $\mathbb{R}$, or the integers, $\mathbb{Z}$ (all of positive, negative and zero). Another nonlogical axiom, for Euclidean geometry, is "Euclid's 5th postulate", which asserts that *through a point outside a line we can draw precisely one parallel to said line.* Again, the existence of so-called non-Euclidean geometries shows that this is not a universal truth.

When we use the term "theory Γ", we mean somewhat ambiguously, but equivalently, either

[28] They express universal truths of *logic*, that is.

[29] Strictly speaking, *formula-forms* or *formula-schemata*, since these formulae will contain, as subformulae, formula-names of unspecified formulae (such as $\mathscr{A}$), arbitrary (object) variables, function names, etc.

- Γ is the set *all its theorems*, that is, the set of all formulae that can be proved starting from the axioms of the theory repeatedly applying the proof-tools (1)–(3) above.

  Or

- Γ is identified with the set of *all the postulated nonlogical axioms*. Clearly, if we have the nonlogical axioms, then using the logical axioms that are common to all theories, along with the proof-process (1)–(3) above, we may write down, in principle,[30] all the theorems of the theory.

We prefer the viewpoint of the second bullet, as it gives prominence to our start-up assumptions; the nonlogical axioms.

The terminology "$\mathscr{F}$ *is true in the theory*" simply means that $\mathscr{F}$ is a *theorem of the theory*. Its truth is *relative to the truth of the nonlogical axioms*; it is not absolute or universal. For example, "the sum of the angles of any triangle equals 180°" is true in (is a theorem of) Euclidean geometry. It is not true in (is not a theorem of) either Riemann's or Lobachevski's geometries.

That $\mathscr{F}$ is true in a theory Σ will be denoted symbolically as $\Sigma \vdash \mathscr{F}$.

Note that the logical axioms are not mentioned at the left of "$\vdash$". Thus, if Σ is empty and we have proved $\mathscr{F}$ *only using logical axioms*, then we will write $\vdash \mathscr{F}$.

It is immediate from the foregoing that since a proof is not obliged, in an applications of step (1) (1.1.1.34), to use any nonlogical axiom, that every theory also contains among its theorems all the absolute truths $\mathscr{F}$ for which $\vdash \mathscr{F}$.[31]

It is clear that $\vdash$ is *transitive*, that is, if we have $\Sigma \vdash \mathscr{A}_i$ for $i = 1, \ldots, n$, and also $A_1, \ldots, A_n \vdash \mathscr{B}$, then $\Sigma \vdash \mathscr{B}$. Indeed, we can clearly concatenate the proofs of each A_i—in any order—and then append the proof of $\mathscr{B}$ at the very end. What we get is a proof of $\mathscr{B}$ as required.

Since at each step of writing a proof we look back rather than forward [steps (2) and (3)], it is clear that chopping off the "tail" of a proof at any point leaves us with a proof. This observation entails that when we attempt to prove a formula from given axioms, we just stop as soon as we have written the formula down. □

1.1.1.36 Exercise. Elaborate on the remark above regarding the transitivity of $\vdash$. □

1.1.1.37 Exercise. In mathematical practice we are allowed to use in a proof any already proved theorems, in a step of type (1) of 1.1.1.34. Carefully justify this practice in the context of Definition 1.1.1.34. □

Since rules (2) and Gen on 1.1.1.34 preserve truth, and the logical axioms are universally true, then so is any $\mathscr{F}$ for which we have $\vdash \mathscr{F}$. Logicians call the content of this observation *soundness*.

[30] "In principle": The set of theorems of an interesting theory such as arithmetic, Euclidean geometry, or set theory is infinite.

[31] By Gödel's completeness theorem, Gödel (1930), the hedging "for which $\vdash \mathscr{F}$" is redundant.

1.1.1.38 Definition. (Logical Axioms) The usually adopted logical axioms are (forall choices of formulae, variables and terms that appear in them):

(i) All tautologies

(ii) $(\forall x)\mathscr{A}[x] \to \mathscr{A}[t]$ (see conclusion of 1.1.1.33)

(iii) $\mathscr{A}[x] \to (\forall x)\mathscr{A}[x]$, provided x is not free in $\mathscr{A}[x]$

(iv) $(\forall x)(\mathscr{A}[x] \to \mathscr{B}[x]) \to (\forall x)\mathscr{A}[x] \to (\forall x)\mathscr{B}[x]$

(v) $x = x$

(vi) $t = s \to (\mathscr{A}[t] \equiv \mathscr{A}[s])$ (see conclusion of 1.1.1.33). □

That the above logical axioms are *adequate* to prove *all* universal truths using the proof mechanism of 1.1.1.34 is a result of Gödel's [completeness theorem; Gödel (1930)].

1.1.1.39 Remark. It is easy to verify that all the logical axioms are indeed universal truths. For group (i) and (v) this is trivial. The "truth" expressed (codified) in group (ii) is that "if $\mathscr{A}[x]$ is true *for all objects in its domain*, then it must be true if we take x to be a specific object t". Note that even if t has input variables, then as they vary over the domain they generate objects from the domain. The generated objects are part of "*all objects in its domain*".

The truth of all formulae in group (iii) follows from a trivial modification of the argument in 1.1.1.23.

Group (vi) is Leibniz's characterization of equality: It states that replacing "equals by equals" in an argument slot (x of of $\mathscr{A}[x]$ in our case) produces the same result.

Finally, let us look at group (iv). For simplicity we assume that x is the only variable, so we will show the truth of $(\forall x)(\mathscr{A}(x) \to \mathscr{B}(x)) \to (\forall x)\mathscr{A}(x) \to (\forall x)\mathscr{B}(x)$ in its domain. First off, this means

$$(\forall x)(\mathscr{A}(x) \to \mathscr{B}(x)) \to \Big((\forall x)\mathscr{A}(x) \to (\forall x)\mathscr{B}(x)\Big) \tag{1}$$

Since (1) is an implication, 1.1.1.14 indicates that the only real work for us is if $(\forall x)(\mathscr{A}(x) \to \mathscr{B}(x))$ evaluates as **t**. If this is so, this means

$$\text{For every } k \text{ in the domain, } \mathscr{A}(k) \to \mathscr{B}(k) \text{ is } \mathbf{t} \tag{2}$$

We now try to prove that the right hand side of the leftmost $\to$ must evaluate as **t**. As it too is an implication, we will only consider the real work case where $(\forall x)\mathscr{A}(x)$ is true, that is

$$\text{For every } k \text{ in the domain, } \mathscr{A}(k) \text{ is } \mathbf{t} \tag{3}$$

and try to obtain that

$$(\forall x)\mathscr{B}(x) \tag{4}$$

is true. Now, via 1.1.1.14, (2) and (3) give "For every k in the domain, $\mathscr{B}(k)$ is **t**", which establishes (4). □

1.1.1.40 Example. Here are some proofs written in extreme pedantry.

(I) Establish the (universal) truth of $\mathscr{A}[t] \to (\exists x)\mathscr{A}[x]$. The proof follows:

(1)	$(\forall x)\neg\mathscr{A}[x] \to \neg\mathscr{A}[t]$	⟨axiom (ii)⟩
(2)	$\mathscr{A}[t] \to \neg(\forall x)\neg\mathscr{A}[x]$	⟨(1) and rule (2) of proof-writing; 1.1.1.34⟩
(3)	$\mathscr{A}[t] \to (\exists x)\mathscr{A}[x]$	⟨(2) and rule (2) of 1.1.1.34, using 1.1.1.20⟩

The comments in ⟨. . .⟩-brackets explain why we wrote the formula to their left. The numbering to the left allows easy reference to previous formulae. The last step is "replacing equivalents by equivalents" in a Boolean combination. The result stays the same since it is as if we "called" $\mathscr{A} \to \mathscr{X}$ with inputs, first $\neg(\forall x)\neg\mathscr{A}[x]$ and then $(\exists x)\mathscr{A}[x]$. But these inputs have the same (truth) value, so both calls will return the same answer. Since step (2) has written a truth (why?), so has step (2).

(II) Establish that $t = t$ for any term t. The proof follows:

(1)	$x = x$	⟨axiom (v)⟩
(2)	$(\forall x)x = x$	⟨(1) + Gen⟩
(3)	$(\forall x)x = x \to t = t$	⟨axiom (ii)⟩
(4)	$t = t$	⟨(2, 3) + tautological implication⟩

□

1.1.1.41 Remark. In the second proof above we used two important tools explicitly. We identify both here so they can be used "off the shelf" in diverse situations in the future.

The step from (2) to (4) via (3) generalizes to the rule "from (the truth of) $(\forall x)\mathscr{A}[x]$ follows (the truth of) $\mathscr{A}[t]$". This rule is called *specialization* or *Spec*. It follows from an application of this tautological implication, $\mathscr{F}, \mathscr{F} \to \mathscr{G} \models_{taut} \mathscr{G}$ known as *modus ponens*, for short MP. The reader can easily verify that indeed MP is a tautological implication, so it qualifies as a proof-step of type (2) (1.1.1.34). □

1.1.1.42 Exercise. Give a proof that from the truth of $\mathscr{A}[x]$ follows the truth of $\mathscr{A}[t]$. □

1.1.1.43 Example. We verify that with an assumption (nonlogical!) of the form $\mathscr{A} \to \mathscr{B}$ we can prove $\mathscr{A} \to (\forall x)\mathscr{B}$, on the proviso that x is not free in $\mathscr{A}$. That is, we verify, under the stated condition, that $\mathscr{A} \to \mathscr{B} \vdash \mathscr{A} \to (\forall x)\mathscr{B}$.

(1)	$\mathscr{A} \to \mathscr{B}$	⟨hyp⟩
(2)	$(\forall x)(\mathscr{A} \to \mathscr{B})$	⟨(1) + Gen⟩
(3)	$(\forall x)(\mathscr{A} \to \mathscr{B}) \to (\forall x)\mathscr{A} \to (\forall x)\mathscr{B}$	⟨axiom (iv)⟩
(4)	$(\forall x)\mathscr{A} \to (\forall x)\mathscr{B}$	⟨(2, 3) + MP⟩

$$(5) \quad \mathscr{A} \to (\forall x)\mathscr{B} \qquad \langle(*)\rangle$$

In step (1) we said "hyp". The formula is a "hypothesis"—a starting point; not something that we claim as true; a nonlogical axiom. In step (5) I wrote (*) so that I can explain the reasons for the step outside the proof, since the explanation is long, namely: (5) follows from (4) by replacing "equivalents for equivalents"—see 1.1.1.23 and the similar situation in 1.1.1.40. □

1.1.1.44 Exercise. Prove that $\mathscr{A} \to \mathscr{B} \vdash \neg\mathscr{B} \to \neg\mathscr{A}$. The two sides of $\vdash$ are called *contrapositives* of each other. □

1.1.1.45 Exercise. Prove that $\mathscr{A} \to \mathscr{B} \vdash (\exists)\mathscr{A} \to \mathscr{B}$ as long as x is not free in $\mathscr{B}$.
Hint. Rely on 1.1.1.44 and use 1.1.1.43. □

1.1.1.46 Example. Let us establish the familiar commutativity property of equality as a result of the logical axioms [in particular, of (v) and (vi); cf. p. 21].

Let $\mathscr{A}[z]$ stand for $z = x$. An instance of Leibniz's axiom is $x = y \to (\mathscr{A}[x] \equiv \mathscr{A}[y])$ i.e.,

$$x = y \to (x = x \equiv y = x) \tag{1}$$

We can now embark on a proof:

(a)	$x = y \to (x = x \equiv y = x)$	⟨logical axiom (1)⟩
(b)	$x = x \to x = y \to y = x$	⟨tautological implication of (a)⟩
(c)	$x = x$	⟨logical axiom⟩
(d)	$x = y \to y = x$	⟨(b, c) + MP⟩

Step (b) takes some doing, but is easy. Recall 1.1.1.29 and 1.1.1.31. We need to argue that if line (a) is **t**, then this forces line (b)

$$x = x \to (x = y \to y = x) \tag{2}$$

to be **t** as well. By the way, the Boolean variables here are $x = x, x = y$ and $y = x$.

Well, the real work toward seeing that (2) is **t** is when $x = x$ and $x = y$ are **t**. If so, the assumption that line (a) is true forces $x = x \equiv y = x$ to be true (because the part to the left of $\to$ in said line is). Since $x = x$ is assumed **t**,[32] then so must $y = x$, which establishes (2).

Since the above proof contains no nonlogical axioms, we may write $\vdash x = y \to y = x$.

The reader will note that this is not a tautology, since $x = y$ and $y = x$ are distinct Boolean variables. □

[32]The word "assumed" was inserted for emphasis: $x = x$ in this argument is a Boolean variable. We are not looking for its intrinsic value, rather we are taking turns to consider each of its "possible" values, **f** and **t**. The argument skipped the first value because it trivially makes (2) true.

1.1.1.47 Exercise. Armed with the commutativity of =, prove the transitivity of =. That is, establish the claim $\vdash x = y \to y = z \to x = z$. □

1.1.1.48 Example. There are a couple of trivial, but often-used proof tools. They are expressed in the form $\mathscr{X}, \mathscr{Y}, \ldots \vdash \mathscr{A}$, that is, "if I know that $\mathscr{X}, \mathscr{Y}, \ldots$ hold—either because they are assumptions (e.g., could be nonlogical axioms) or are already proved theorems—then I can prove that $\mathscr{A}$ holds as well".

(a) **Proof by cases.** $\mathscr{A} \to \mathscr{B}, \mathscr{C} \to \mathscr{B} \vdash \mathscr{A} \vee \mathscr{C} \to \mathscr{B}$.

It states that to prove that $\mathscr{B}$ follows from a disjunction, it *suffices* to prove separately that $\mathscr{B}$ follows from each *case*—$\mathscr{A}$ and $\mathscr{C}$—of the disjunction.

(b) **Ping pong.** $\mathscr{A} \to \mathscr{B}, \mathscr{B} \to \mathscr{A} \vdash \mathscr{A} \equiv \mathscr{B}$. It states that to prove an equivalence, $\mathscr{A} \equiv \mathscr{B}$, it *suffices* to prove *each direction*—$\mathscr{A} \to \mathscr{B}$ and $\mathscr{B} \to \mathscr{A}$—separately, since, from the two directions taken as hypotheses jointly, we can prove the equivalence.

Each of (a) and (b) admit immediate proofs: Once we have assumed the hypotheses on each side, a tautological implication yields the conclusion at once [cf. 1.1.1.34, rule (2) applied]. □

A major proof tool of the mathematician and computer scientist is the so-called *deduction theorem*. It is stated without proof here—for us it is its statement that matters.

See Tourlakis (2003a) or Tourlakis (2008) for a proof of the deduction theorem, but be warned that the two versions in these references are different, because the specific *foundations* of logic in these two are different! The difference lies in how generalization is applied, and that affects the statement, and proof, of the deduction theorem. The present volume uses the style of generalization as it is practiced in the first cited reference.

1.1.1.49 Theorem. (Deduction Theorem) *If we can prove $\mathscr{B}$ from assumptions Γ and $\mathscr{A}$, then we can prove $\mathscr{A} \to \mathscr{B}$ from the assumptions Γ alone,* on a condition. *The condition is that during the proof of $\mathscr{B}$ from hypotheses Γ and $\mathscr{A}$, step (3) of 1.1.1.34 was never applied with a variable that occurs free in $\mathscr{A}$.*

In other words, the free variables of $\mathscr{A}$ during said proof are "frozen"; they behave like constants.

We can say the above symbolically as "if $\Gamma, \mathscr{A} \vdash \mathscr{B}$, then $\Gamma \vdash \mathscr{A} \to \mathscr{B}$, on a condition, etc."

What is the deduction theorem good for? Well, for one thing, it tells us that instead of proving $\mathscr{A} \to \mathscr{B}$ it suffices to prove the less complex—since the glue $\to$ and the part $\mathscr{A}$ are removed—$\mathscr{B}$. For another, we have the added bonus of "knowing more" before starting the proof. While toward proving $\mathscr{A} \to \mathscr{B}$ we "knew" Γ, toward proving $\mathscr{B}$ we also know $\mathscr{A}$.

Is the restriction on $\mathscr{A}$ too limiting? Not really. In practice, say, we want to prove that $\Gamma \vdash \mathscr{A}(x,y) \to \mathscr{B}(z,y,w)$. We proceed as follows:

> *Fix all the variables* of $\mathscr{A}$, here x and y, to some unspecified values. *Remember not to use either x or y in a rule Gen during the proof!*
>
> Assume now $\mathscr{A}(x,y)$. Proceed to prove $\mathscr{B}(z,y,w)$—recall that y is a "constant".
>
> By the deduction theorem, we have proved $\mathscr{A}(x,y) \to \mathscr{B}(z,y,w)$ from just Γ, because we never performed generalization with the frozen x and y.

In practice we apply less pedantry in the process:

(1) We only say, "let us fix the values all free variables $x, y, \ldots$ in $\mathscr{A}$".

We are, of course, obliged to remember that *this also fixes these same variables in $\mathscr{B}$, and everywhere else, throughout the proof.* Thus, we *cannot* universally quantify any of these variables during the proof, nor can we (a subsidiary operation this; cf. 1.1.1.42) substitute a term into such "frozen" variables.

(2) The task ends as soon as we prove $\mathscr{B}$.

We do *not* need to repeat this kind of justification every time: "By the deduction theorem, we have proved $\mathscr{A} \to \mathscr{B}$ from just Γ, etc."

We will see this proof technique applied many times in this book, starting from the next subsection.

We conclude with the ancient technique of *proof by contradiction.*[33] We will define a *contradiction* to be a formula of the form $\mathscr{A} \wedge \neg\mathscr{A}$. From the truth tables we know that this evaluates as **f** regardless of whether $\mathscr{A}$ itself evaluates a **t** or **f**. The reader can verify at once, that for any $\mathscr{F}$, $\mathscr{A} \wedge \neg\mathscr{A} \models_{taut} \mathscr{F}$.

1.1.1.50 Theorem. (Proof by Contradiction) *For any closed $\mathscr{A}$, we have $\Gamma \vdash \mathscr{A}$ iff $\Gamma, \neg\mathscr{A} \vdash \mathscr{X} \wedge \neg\mathscr{X}$, for some $\mathscr{X}$.*

Proof. Indeed, for the *if*-part, let $\Gamma, \neg\mathscr{A} \vdash \mathscr{X} \wedge \neg\mathscr{X}$. By the deduction theorem (applicable without hedging as $\mathscr{A}$ is closed) we get

$$\Gamma \vdash \neg\mathscr{A} \to \mathscr{X} \wedge \neg\mathscr{X} \tag{1}$$

It is straightforward to see that $\neg\mathscr{A} \to \mathscr{X} \wedge \neg\mathscr{X} \models_{taut} \mathscr{A}$, hence by transitivity of $\vdash$, we get $\Gamma \vdash \mathscr{A}$.

only if-part. Say, $\Gamma \vdash \mathscr{A}$. Adding to the assumptions Γ we can still prove $\mathscr{A}$ (see that you agree! 1.1.1.34). Thus

$$\Gamma, \neg\mathscr{A} \vdash \mathscr{A} \tag{2}$$

[33] Often used by Euclid, for example.

But also (why?)

$$\Gamma, \neg\mathscr{A} \vdash \neg\mathscr{A} \tag{3}$$

The trivial $\mathscr{A}, \neg\mathscr{A} \models_{taut} \mathscr{A} \wedge \neg\mathscr{A}$ along with (2) and (3) above, and transitivity of $\vdash$, yield $\Gamma, \neg\mathscr{A} \vdash \mathscr{A} \wedge \neg\mathscr{A}$. □

The technique is used as follows: To prove $\Gamma \vdash \mathscr{A}$ (closed $\mathscr{A}$) we start by "Assume, by way of contradiction, $\neg\mathscr{A}$" and then proceed to indeed obtain one.

1.1.1.51 Definition. A mathematical theory, given by its set of nonlogical axioms, is *consistent* or *free from contradiction*, provided it is impossible to prove a contradiction from its axioms. Otherwise it is called inconsistent. □

Thus we can rephrase 1.1.1.50 as $\Gamma \vdash \mathscr{A}$ iff (the theory with axioms) $\Gamma, \neg\mathscr{A}$ is inconsistent.

1.1.1.52 Remark. ("Everyday" Proof Style) A few important remarks are in order to conclude our digression into logic.

(1) The proof of truth of a formula using *first principles* from 1.1.1.13 and working *directly* with a reference set is now for us a thing of the past. The last time we utilized the method was to establish that all the logical axioms were indeed universal truths (1.1.1.39—see also 1.1.1.33). From then on we will ride on the shoulders of our *logical axioms* 1.1.1.38, and whatever other assumptions *we take as true* from time to time, to prove all our theorems, essentially "syntactically", that is, by writing proofs according to 1.1.1.34.

(2) The practicing mathematician or computer scientist uses a simplified, often shorter, and rather conversational version of the *proofs with annotation* that we presented so far (for example, in 1.1.1.43 and 1.1.1.46). We should get used to this relaxed style. Here is an example. We will establish that

$$\vdash (\forall x)(\mathscr{A} \wedge \mathscr{B}) \equiv (\forall x)\mathscr{A} \wedge (\forall x)\mathscr{B}$$

Proof. We employ ping pong. $(\rightarrow)$ direction: Assume $(\forall x)(\mathscr{A} \wedge \mathscr{B})$ with all its free variables frozen (we are going via the deduction theorem). Remove the quantifier (Spec) to obtain $\mathscr{A} \wedge \mathscr{B}$ and apply two tautological implications to obtain $\mathscr{A}$ and $\mathscr{B}$. Apply Gen to each to get $(\forall x)\mathscr{A}$ and $(\forall x)\mathscr{B}$. A tautological implication yields what we want.

For the $(\leftarrow)$ direction, assume $(\forall x)\mathscr{A} \wedge (\forall x)\mathscr{B}$, freezing all free variables of the formula. Two tautological implications yield $(\forall x)\mathscr{A}$ and $(\forall x)\mathscr{B}$. Two applications of Spec, followed by tautological implication yield $\mathscr{A} \wedge \mathscr{B}$. Via Gen we get what we want.

(3) We finally establish that $\mathscr{A} \rightarrow \mathscr{B} \vdash (\forall x)\mathscr{A} \rightarrow (\forall x)\mathscr{B}$. Indeed, the hypothesis yields $(\forall x)(\mathscr{A} \rightarrow \mathscr{B})$ by Gen. We are done via axiom (iv) (1.1.1.38) and modus ponens. □

1.1.1.53 Exercise. Establish the fact $\mathscr{A} \equiv \mathscr{B} \vdash (\forall x)\mathscr{A} \equiv (\forall x)\mathscr{B}$.
Hint. Use 1.1.1.52. □

1.1.1.54 Exercise. Establish the fact $\mathscr{A} \equiv \mathscr{B} \vdash (\exists x)\mathscr{A} \equiv (\exists x)\mathscr{B}$.
Hint. Use 1.1.1.53. □

1.1.1.55 Exercise. Establish the fact $\vdash (\exists x)(\mathscr{A} \vee \mathscr{B}) \equiv (\exists x)\mathscr{A} \vee (\exists x)\mathscr{B}$.
Hint. Use 1.1.1.52. □

1.1.2 Sets and their Operations

We now return to our brief study of sets. The naïve set theory of Cantor is not axiomatic. In our very brief and elementary review of it we will deviate only slightly and adopt precisely one axiom. First off, consider the formulae $x \in A$ and $x \in B$. By 1.1.1.38(vi), we obtain[34]

$$A = B \to (x \in A \equiv x \in B)$$

and further, via 1.1.1.43 we get (since x, A, B are distinct variables)

$$A = B \to (\forall x)(x \in A \equiv x \in B) \tag{1}$$

Suppose next that A and B stand for sets*—no such restrictive assumption was made above.* Then (1) says that if two sets are equal, then every member of one (x) is a member of the other, and vice versa; they have precisely the same elements. Is the converse[35] true?

That it is so is a fundamental property of sets; a nonlogical axiom. It is the so-called *axiom of extensionality*.

For any sets A and B, $(\forall x)(x \in A \equiv x \in B) \to A = B$ $\qquad (Ext)$

Extensionality says that the extension—what sets contain— is what matters to determine their equality. In particular, "structure" does not matter. Nor does "intention": e.g., whether we say outright, "collect 1 and 2 into a set", or, in a roundabout way, "collect the roots of the equation $x^2 - 3x + 2 = 0$ into a set" we get the same set.

Taking (1) and (Ext) together (ping pong) we have [cf. (b) in 1.1.1.48]:

For any sets A and B, $A = B \equiv (\forall x)(x \in A \equiv x \in B)$ $\qquad (2)$

Since $x \in 2$ and $x \in 3$ are false, as 2 and 3 are numbers and thus contain no elements,

$$(\forall x)(x \in 2 \equiv x \in 3) \to 2 = 3$$

is false, since to the left of $\to$ we have a true formula, while to the right a false one. Thus, the restriction on the *type* of A and B in (Ext) and (2) is essential. Of course, (1) is *valid for any type of variables* A, B, x.

[34]The mathematician and computer scientist will rather say "we obtain $\mathscr{X}$" to indicate he proved so, without using the provability symbol $\vdash$. He will also let the context fend for itself as to what the assumptions were; here no nonlogical assumptions were made.
[35]The converse of the implication $\mathscr{A} \to \mathscr{B}$ is $\mathscr{B} \to \mathscr{A}$.

1.1.2.1 Example. Let us introduce the notation-by-listing of sets, $\{\ldots\}$, where the "..." is in each case replaced by an explicit listing of all the set members. Here are two examples: $\{1, 2, 1, 2, 2\}$ and $\{2, 1\}$. These two sets are equal by inspection, according to extensionality, for every element of one appears in the other, and vice versa. In particular, *neither multiplicity, nor order of listing matter*. Only the presence of an element matters, not where (in the listing), or how many times. So, we may write $\{1, 2, 1, 2, 2\} = \{2, 1\}$ or indeed $\{2, 1\} = \{1, 2, 1, 2, 2\}$.

It should be clear that only the (intuitively) *finite sets* (a concept we will soon mathematically define) can be depicted by listing; and this only in principle, since we may not want to list a set of one trillion elements. By the way, writing $\mathbb{N} = \{0, 1, 2, \ldots\}$ is not a by-listing depiction of $\mathbb{N}$, rather, it is a sloppy and *abused* notation (yet surprisingly common). The "..." indicates an unending, and understood from the *context*, *process* whereby the next element is *generated* by adding one to the previous. The notation taken out of context is nonsensical and gives no clue as to what "..." means. □

The notation $A \subseteq B$, read "A is a *subset* of B", or "B is a *superset* of A", means that every member of A is in B as well. So it is given by the mathematical definition below:

$$A \subseteq B \stackrel{\text{Def}}{\equiv} (\forall x)(x \in A \to x \in B) \tag{3}$$

So how does one prove, given some sets A and B that $A \subseteq B$? One uses definition (3) above, and proves instead $(\forall x)(x \in A \to x \in B)$. But to so prove, it suffices to prove instead $x \in A \to x \in B$, since an application of Gen to this formula produces the preceding one.

Perfect! One can then proceed as follows: "Fix x and assume $x \in A$". All that one needs to do next is to prove $x \in B$ (1.1.1.49).

At the intuitive level, and from the "word description of equality and subset relations", we expect, for sets A and B, that if $A = B$, then also $A \subseteq B$ (and by symmetry, also $B \subseteq A$). This can be mathematically proved as well: The assumption means $(\forall x)(x \in A \equiv x \in B)$. Dropping $(\forall x)$ (Spec) and following up with a tautological implication we get $x \in A \to x \in B$. Reintroducing $(\forall x)$ (Gen) we get $A \subseteq B$ (Definition (3)).

Intuitively, for any to sets A and B, if we know that $A \subseteq B$ and $B \subseteq A$, then $A = B$ (the vice versa was the content of the preceding paragraph). Indeed, the two assumptions and (3) above expand to $(\forall x)(x \in A \to x \in B)$ and $(\forall x)(x \in B \to x \in A)$, respectively. Dropping $(\forall x)$ (Spec) we obtain $x \in A \to x \in B$ and $x \in B \to x \in A$. Following up with a tautological implication we get $x \in A \equiv x \in B$. Applying $(\forall x)$ we get $A = B$.

So, in practice, to prove set equality, $A = B$, we go about it like this: "$(\subseteq)$ direction: Fix $x \in A \ldots$ therefore, $x \in B$ is proved". Then we do: "$(\supseteq)$ direction: Fix $x \in B \ldots$ therefore, $x \in A$ is proved".

If $A \subseteq B$ but $A \neq B$ we say that "A is a *proper subset* of B" and write $A \subset B$. As is usual in mathematics, negating a relation is informally denoted by the relation

symbol superimposed with a "/". So $A \notin B$, $A \not\subseteq B$ and $A \not\subset B$ mean $\neg A \in B$, $\neg A \subseteq B$ and $\neg A \subset B$, respectively.

1.1.2.2 Definition. (Bounded Quantification) In much of what we do in this volume we will find *bounded quantifiers* very useful. That is, in set theory we will often want to says things like "for all x in A, $\mathscr{X}[x]$ holds". While this can be captured by $(\forall x)(x \in A \to \mathscr{X}[x])$, the alternative shorthand is much in use, and preferable: $(\forall x \in A)\mathscr{X}[x]$ or also $(\forall x)_{\in A}\mathscr{X}[x]$.

In arithmetic we will correspondingly often want to say "for all $x < y$, $\mathscr{X}[x]$ holds". This is coded directly as $(\forall x)(x < y \to \mathscr{X}[x])$. The preferred shorthand is: $(\forall x < y)\mathscr{X}[x]$ or also $(\forall x)_{<y}\mathscr{X}[x]$. In these two cases, respectively, A and y are free variables. □

1.1.2.3 Remark. The corresponding "for some x in A, $\mathscr{X}[x]$ holds" and "for some $x < y$, $\mathscr{X}[x]$ holds" have the shorthands $(\exists x \in A)\mathscr{X}[x]$ or $(\exists x)_{\in A}\mathscr{X}[x]$ for the former and $(\exists x < y)\mathscr{X}[x]$ or also $(\exists x)_{<y}\mathscr{X}[x]$ for the latter.

The shorthand $(\exists x \in A)\mathscr{X}[x]$ and $(\exists x < y)\mathscr{X}[x]$ stand for $(\exists x)(x \in A \wedge \mathscr{X}[x])$ and $(\exists x)(x < y \wedge \mathscr{X}[x])$, respectively.

Translating to $\forall$ notation we do not get any nasty surprises. For example,

$$(\exists x)(x \in A \wedge \mathscr{X}[x]) \qquad (*)$$

is equivalent to $\neg(\forall x)\neg(x \in A \wedge \mathscr{X}[x])$. Using 1.1.1.53 and an obvious tautology, we see that this is the same as $\neg(\forall x)(x \in A \to \neg X[x])$; in shorthand: $\neg(\forall x \in A)\neg X[x]$. Neat! The original $(*)$ has the bounded-quantifier expression $(\exists x \in A)\mathscr{X}[x]$, so the "$\exists \equiv \neg\forall\neg$" property (1.1.1.20) holds for bounded quantifiers! □

1.1.2.4 Exercise. Show that the "$\forall \equiv \neg\exists\neg$" property (1.1.1.21) holds for bounded quantifiers. □

There is a more general way to build sets than by just collecting together and listing a finite number of elements; by "defining property". That is, for any formula $\mathscr{A}(x)$ we collect into a set all the x (values) for which $\mathscr{A}(x)$ is true. We denote this set by the term $\{x : \mathscr{A}(x)\}$. Of course, $\mathscr{A}(x)$ is the defining property or "entrance requirement" that determines membership. To make this precise we define

1.1.2.5 Definition. $S = \{x : \mathscr{A}(x)\}$ is shorthand, suggestive, notation for $(\forall x)(x \in S \equiv \mathscr{A}(x))$. □

1.1.2.6 Remark. Several remarks are in order.

(1) The S that enters in $(\forall x)(x \in S \equiv \mathscr{A}(x))$ is *unique*, that is, if also $(\forall x)(x \in T \equiv \mathscr{A}(x))$, then $S = T$. Indeed, the two imply (Spec) $x \in S \equiv \mathscr{A}(x)$ and $x \in T \equiv \mathscr{A}(x)$, thus, $x \in S \equiv x \in T$ by tautological implication. Generalizing we get $(\forall x)(x \in S \equiv x \in T)$, and hence $S = T$.

Pause. Why not just say "The two mean $S = \{x : \mathscr{A}(x)\}$ and $T = \{x : \mathscr{A}(x)\}$, hence $S = T$ by transitivity of equality"?◀

Because the notation "$S = \{x : \mathscr{A}(x)\}$" is only shorthand for something else. The symbol of equality "=" is inserted in *anticipation*, but *not* due to an *a priori* knowledge, that it will behave correctly, *as equality*. Now we know—through the longer argument and *post facto*—that it was all right to have written "=" after all.

(2) Renaming the bound variable of $(\forall x)(x \in S \equiv \mathscr{A}(x))$ into (a new) z we get the equivalent formula (cf. 1.1.1.33) $(\forall z)(z \in S \equiv \mathscr{A}(z))$. The latter says $S = \{z : \mathscr{A}(z)\}$. So, x (and z) in $\{x : \mathscr{A}(x)\}$ is a bound variable that can be renamed without changing the meaning.

(3) By specializing $(\forall x)(x \in S \equiv \mathscr{A}(x))$ we get $t \in S \equiv \mathscr{A}(t)$ for any term t. This says what our intuition wants: To test if an object t is in the set S, just test that it passes the entrance requirement: $\mathscr{A}(t)$.

Another way to say the same thing is $t \in \{x : \mathscr{A}(x)\}$ iff $\mathscr{A}(t)$.

(4) It is time to be reminded (this was mentioned in passing earlier) that it is *not* the case that every formula $\mathscr{A}(x)$ leads to a *set* $\{x : \mathscr{A}(x)\}$. To think so leads to nasty contradictions, as it did in Cantor's naïve set theory. Examples of formulae that are *not* set-builders are $x \notin x$ and $x = x$ (cf. Section 1.3).

(5) The statements $\{x : \mathscr{A}[x]\} = \{x : \mathscr{B}[x]\}$ and $\mathscr{A}[x] \equiv \mathscr{B}[x]$ are equivalent. Indeed, if we assume the first, then by (1) on p. 27 [which is no more than an application of (vi) from 1.1.1.38] we get $x \in \{x : \mathscr{A}[x]\} \equiv x \in \{x : \mathscr{B}[x]\}$, which by (3) above is (replacing equivalents by equivalents) $\mathscr{A}[x] \equiv \mathscr{B}[x]$. These steps can be reversed [in this direction Ext is invoked rather than (1) on p. 27] to prove the converse.

This is not unexpected at an intuitive level, but it is nice to have it affirmed mathematically: If two "defining properties" are equivalent, then they yield the same result, true or false, on every object we apply them. Thus, precisely the same objects will "pass" each of the two.

(6) $\{x : \mathscr{A}(x, y, z, \ldots)\}$ denotes several different sets (modulo the previous warning), one for each choice of the unspecified values $y, z, \ldots$. These $y, z, \ldots$ are called *parameters*. □

1.1.2.7 Example. A few paragraphs ago we called the set-building process by defining property "more general" than the process of building sets by grouping members and listing them explicitly between braces { }. Here is why: $\{a, b\} = \{x : x = a \vee x = b\}$. This simple "trick" can be applied to any finite set, to represent it by a defining property, namely, the disjunction of atomic formulae of the form $x = a$ for every a that we want to include in the set. □

There is nothing in naïve set theory that helps us argue that the collection of just two objects—the so-called (unordered) *pair*—is a set (one that does not lead to contradictions, that is).[36] In the naïve approach we take it for granted as a "self evident" fact! Equipped with the hindsight of the early (naïve) set theory paradoxes and their workarounds, we can be content that a pair is so "small" as a collection—just two elements—and is not about to cause any problems. In axiomatic approaches there is an axiom which says that we can indeed form a *set* of two elements.

1.1.2.8 Example. $\{x : x \in \mathbb{N} \wedge x > y\}$ consists of all numbers in $\mathbb{N}$ greater than y. That is: $y+1, y+2, y+3, \ldots$ We may also use the (abuse of) notation $\{x \in \mathbb{N} : x > y\}$ for this set. □

1.1.2.9 Example. Sometimes we collect more complicated objects than values of variables. For example, $\{x^2 : x = 2 \vee x = 9\}$ is the set $\{2^2, 9^2\}$, i.e., $\{4, 81\}$.

A more complicated example is $S = \{x+y : 0 < x < y\}$, where x, y are varying over $\mathbb{N}$.

(1) Suppose that y is the only a parameter. Then $S = \{y+1, y+2, \ldots, 2y-1\}$.

(2) Suppose x is the only a parameter. Then $S = \{2x+1, 2x+2, \ldots\}$. This is all of $\mathbb{N}$, except the segment from 0 to $2x$.

(3) Suppose neither of x or y are parameters. Then $S = \{3, 4, 5, \ldots\}$.

(4) Finally, suppose that both x and y are parameters. Then what $S = \{x+y : 0 < x < y\}$ denotes is an infinite family of one-element sets, using all the elements of $\mathbb{N}$ except $0, 1, 2$: $\{3\}, \{4\}, \{5\}, \ldots$ □

Because of (4) in 1.1.2.6, mathematicians found a way to *limit the size of collections* to ensure they are, technically, sets. An easy (but not the only) way to do this is to build any new sets as parts (subsets) of some other set that we have already built.

Thus all our discussions in set theory will have some—usually unspecified, large enough to be useable but not too large to be troublesome; and totally unobtrusive[37]—*reference set* tucked away somewhere; let us call it $\mathbf{U}$.

This $\mathbf{U}$ is our "resource" where we take our *set theory* objects from, give values to our variables from, and have our quantifiers vary over. Thus, in set *naïve* theory, when we write $(\forall x)$ or $(\exists x)$ we really mean $(\forall x \in \mathbf{U})$ or $(\exists x \in \mathbf{U})$, respectively. This reference set that we put aside for a discussion is also called the *domain* (of *discourse*).

In other branches of mathematics whose objects can be collected into a set we are less vague about the reference set; thus the calculus of one variable has $\mathbb{R}$ as its domain, while Peano arithmetic has $\mathbb{N}$ as its domain.

It is convenient to have a set with no elements around, the "empty set". This is the set S given by the condition

$$(\forall x)(x \in S \equiv x \neq x\} \qquad (*)$$

[36] Let's face it: naïve set theory is not exactly clear as to what a set is, nor does it care.

[37] We do not pin it down.

By (1) in 1.1.2.6, there is just one set S that satisfies $(*)$, and the symbol reserved for it is $\emptyset$. That it is a set is not in question as it is far too small! It contains nothing. Indeed, by 3 in 1.1.2.6,

$$x \in \emptyset \equiv x \neq x \qquad (**)$$

thus *no x passes the entrance test, the later being false for all x.* Of course, we can write

$$\emptyset = \{x : \neg x = x\}$$

It is useful to note that for any set A, we have $\emptyset \subseteq A$. Indeed, this means $x \in \emptyset \rightarrow x \in A$. This is true since, by $(**)$, $x \in \emptyset$ is false.

If we want to build more *complex* sets we will do well to devise operations on sets. Thus,

1.1.2.10 Definition. If A and B are sets, then their *union*, $A \cup B$, is the set $\{x : x \in A \vee x \in B\}$. □

$A \cup B$ is formed by emptying the members of A and B in a single $\{\ldots\}$ "bag".

The union makes sense even if one or both of A and B stand for atomic elements with no set-theoretic structure, such as numbers.[38] For example, if B is atomic, then $x \in B$ is false, and hence $x \in A \vee x \in B \equiv x \in A$ by 1.1.1.14. That is, $A \cup B = A$ in this case. If both A and B are urelements, then $A \cup B = \emptyset$.

The reader may be wondering: Is it not better to not allow things like $x \in 2$—to make it "illegal", rather than false? No. For one thing, that would mean that before we build an atomic formula $t \in s$ we would then have to analyze first s to ensure it is *not* an urelement; betraying that syntax has to be determined, well, syntactically! Secondly, it would require far too many special cases to be considered in all our definitions.

Thus the following is true [with or without a leading $(\forall x)$; cf. 1.1.1.15]:

$$x \in A \cup B \equiv x \in A \vee x \in B$$

We can form the union of three sets as either $A \cup (B \cup C)$ or $(A \cup B) \cup C$. Since $x \in A \cup (B \cup C)$ is equivalent to

$$x \in A \vee \Big(x \in B \vee x \in C\Big) \qquad (1)$$

and $x \in (A \cup B) \cup C$ is equivalent to

$$\Big(x \in A \vee x \in B\Big) \vee x \in C \qquad (2)$$

The equivalence of (1) and (2) proves that $A \cup (B \cup C) = (A \cup B) \cup C$ which renders brackets irrelevant in a chain of two $\cup$—indeed, in any finite chain of $\cup$ by refining

[38] Such atomic elements are called *urelements* in the literature.

the previous argument (e.g., using induction[39] on the number of $\cup$ symbols in the chain).

1.1.2.11 Exercise. For any sets A and B, prove $A \cup B = B \cup A$. □

1.1.2.12 Example. For any set A, we have $A \cup \emptyset = A$. Indeed, this translates to [using 1.1.1.15 to eliminate $(\forall x)$, and (3) in 1.1.2.6]

$$x \in A \vee x \neq x \equiv x \in A$$

which is clearly true, since $x \neq x$ is false (cf. 1.1.1.14). □

The "big $\cup$" is a very important generalization of union applied to any collection of sets (and/or urelements), not just finitely many.

1.1.2.13 Definition. (Generalized Union) Let S be a set (may contain sets and/or urelements; may be, intuitively, finite or infinite). The symbol $\bigcup S$ denotes the set that we build by emptying the contents of *every set* in S into a new container.

Mathematically,

$$\bigcup S \stackrel{Def}{=} \{x : (\exists A \in S) x \in A\}$$

That is, an "x" is put into the new container iff we can spot it inside *some* A, which in turn is in S. □

We have some special cases of the $\bigcup S$: If S is a collection of sets A_i—$\{A_0, A_1, A_2, \ldots\}$—indexed by $i \in \mathbb{N}$ we may write alternatively

$$\bigcup_{i=0}^{\infty} A_i \text{ or } \bigcup_{i \geq 0} A_i \text{ or } \bigcup_{i \in \mathbb{N}} A_i$$

More generally, we may have a collection of sets A_a indexed by a set I other than $\mathbb{N}$—e.g., $I = \mathbb{R}$, $I = \{2, 3\}$. We indicate $\bigcup S$ in this case by the alternative

$$\bigcup_{a \in I} A_a$$

1.1.2.14 Example. $\bigcup_{a \in \{2,3\}} A_a = A_2 \cup A_3$. □

1.1.2.15 Definition. If A and B are sets, then their *intersection*, $A \cap B$, is the set $\{x : x \in A \wedge x \in B\}$. If $A \cap B = \emptyset$ then we call A and B *disjoint*. □

$A \cap B$ is formed by emptying *only the common* members of A and B in a single $\{\ldots\}$ bag. The intersection makes sense even if one or both of A and B are urelements.

[39] Knowledge of induction, as well as of everything else in this review is presupposed; this is only a *review*! Induction will be our review-subject in Section 1.4.

For example, if B is atomic, then $x \in B$ is false, and hence $x \in A \wedge x \in B$ is false 1.1.1.14. That is, $A \cap B = \emptyset$ in this case.

1.1.2.16 Definition. (Generalized Intersection) Let S be a set (may contain sets and/or urelements; may be, intuitively, finite or infinite). The symbol $\bigcap S$ denotes the set that we build by emptying the contents *that are common to every member* of S into a new container. Mathematically,

$$\bigcap S \stackrel{\text{Def}}{=} \{x : (\forall A \in S) x \in A\}$$

That is, an "x" is put into the new container iff we can spot it inside *every* A in S. □

Thus, if a urelement or $\emptyset$ are members of S, then $\bigcap S = \emptyset$.

For this definition too we have some special cases of the $\bigcap S$: If S is a set of sets $\{A_0, A_1, A_2, \ldots\}$ we may write alternatively

$$\bigcap_{i=0}^{\infty} A_i \text{ or } \bigcap_{i \geq 0} A_i \text{ or } \bigcap_{i \in \mathbb{N}} A_i$$

More generally, we may have a collection of sets A_a indexed by a set I other than $\mathbb{N}$—e.g., $I = \mathbb{R}$, $I = \{0, 1, 2, 3, 11\}$. We indicate $\bigcap S$ in this case by the alternative

$$\bigcap_{a \in I} A_a$$

1.1.2.17 Example. $\bigcap_{a \in \{0,7\}} A_a = A_0 \cap A_7$. □

1.1.2.18 Example. What is $\bigcap \emptyset$? By 1.1.2.16

$$x \in \bigcap \emptyset \equiv (\forall A \in \emptyset) x \in A$$

that is

$$x \in \bigcap \emptyset \equiv (\forall A)(A \in \emptyset \rightarrow x \in A)$$

Since $A \in \emptyset$ is false, the entire right hand side of $\equiv$ is true. That is, the left hand side is true precisely for every x. Recalling that "every x" means "every x-value in the domain $\mathbf{U}$", we have

$$x \in \bigcap \emptyset \equiv x \in \mathbf{U}$$

hence $\bigcap \emptyset = \mathbf{U}$.

Were it not for the "protection" afforded us by the *domain*, "every x" would mean "everything", and we cannot form the *set of everything*! □

1.1.2.19 Definition. If A and B are sets, then their *difference*, $A - B$, is the set $\{x : x \in A \wedge x \notin B\}$. We may also write this as $\{x \in A : x \notin B\}$. If $A = \mathbf{U}$ then we write $\overline{B}$ for $\mathbf{U} - B$ and call it the *complement* of B. □

1.1.2.20 Theorem. $A - B = A \cap \overline{B}$.

Proof. $x \in A - B$ means $x \in A \wedge x \notin B$. Given that $x \notin B \equiv x \in \overline{B}$, we are done. □

1.1.2.21 Example. Let us compute $\{a, b\} - \{a\}$. Now, if $a = b$, then $\{a, b\} = \{a\}$, hence the difference equals $\emptyset$. So let $a \neq b$. We have $\{a, b\} = \{x : x = a \vee x = b\}$ and $\{a\} = \{x : x = a\}$, thus

$$\{a, b\} - \{a\} = \{x : (x = a \vee x = b) \wedge x \neq a\} \tag{1}$$

The reader will have no trouble verifying that, since both $x = a \vee x = b$ and $\neg x = a$ are true in our context, we have

$$(x = a \vee x = b) \wedge \neg x = a \equiv x = b \tag{2}$$

Indeed, in the context of $\{a, b\} - \{a\}$, the truth-value of $x = a$ is **f** and thus the left hand side of $\equiv$ in (2) has the same truth-value as the right hand side—cf. 1.1.1.14 and note the truth-value of $\mathscr{A} \wedge \mathscr{B}$ when $\mathscr{A}$ is **t** and also the truth-value of $\mathscr{A} \vee \mathscr{B}$ when $\mathscr{A}$ is **f**. Therefore the right hand side of (1) simplifies to $\{x : x = b\}$ [cf. 1.1.2.6, item (5)], i.e., $\{b\}$. This is the difference. □

1.1.2.22 Example. What conclusions may we draw from the following equality?

$$\Big\{\{a\}, \{a, b\}\Big\} = \Big\{\{A\}, \{A, B\}\Big\} \tag{1}$$

Well, we get, first off, that

$$\bigcap \Big\{\{a\}, \{a, b\}\Big\} = \bigcap \Big\{\{A\}, \{A, B\}\Big\}$$

by an application of Exercise 1.8.4 (p. 85). That is,

$$\{a\} = \{A\}$$

hence

$$a = A \tag{2}$$

This time let's apply $\bigcup$ to both sides of (1), We get $\{a, b\} = \{A, B\}$, which, by (2), becomes

$$\{a, b\} = \{a, B\} \tag{3}$$

Applying again Exercise 1.8.4 (p. 85), to the function $x - \{a\}$ this time, we get via (3)

$$\{a, b\} - \{a\} = \{a, B\} - \{a\} \tag{4}$$

If $a = b$, then the left hand side of (4) is $\emptyset$, so $a = B$ and therefore

$$b = B \tag{5}$$

If $a \neq b$, then, also $a \neq B$ (else the right hand side, and hence the left, is $\emptyset$). By the previous example, (4) yields $\{b\} = \{B\}$ in this case, so we obtain (5) once more.

In summary, (1) implies (2) and (5). □

1.1.2.23 Definition. (Kuratowski's Ordered Pair) For any objects x and y (sets or not), we reserve the symbol (x, y) as an *abbreviation* of the set $\{\{x\}, \{x, y\}\}$. We call (x, y) the *ordered pair* of x and y. □

The nomenclature for (x, y) stems from the property established in 1.1.2.22, that

$$\text{If } (x, y) = (X, Y), \text{ then } x = X \text{ and } y = Y \qquad (pair)$$

that is, *order* or *position* matters in the pair. This property is not shared by $\{a, b\}$ since, by extensionality, $\{a, b\} = \{b, a\}$, as we know. Of course, the converse—that $x = X$ and $y = Y$ implies $(x, y) = (X, Y)$—is not miraculous at all, and simply follows by two applications of Exercise 1.8.4, p. 85: First, $(x, z) = (X, z)$ and then $(x, y) = (X, Y)$.

The reader is familiar with ordered pairs from analytic geometry, where ordered pairs of real numbers give the coordinates of points on the Cartesian plane. Indeed, the concept of Cartesian product relies on the (x, y) objects.

1.1.2.24 Example. So, $(1, 2) \neq (2, 1)$ lest $1 = 2$. Is $(1, 1) = \{1\}$? To ask explicitly, is $\{\{1\}, \{1, 1\}\} = \{1\}$—that is, by extensionality, twice—is $\{\{1\}\} = \{1\}$? Not unless $\{1\} = 1$, but this cannot be since 1 is an urelement, it has no set-theoretic structure. □

How about

$$A = \{A\} \qquad (*)$$

in general, for some set A? We cannot use here a "type" argument as we did above, since both sides of = are of type set.

Can this be? An unfair question this, since naïve set theory cannot resolve it. If we grant $(*)$, then we have $A \in A$. Well, can *this* be?

Axiomatic foundations disallow this state of affairs, basing it on an intuitive concept[40] that "sets are formed by stages", so you can't have a set (built) before you have (built) its members. $A \in A$ requires the left A being available before the right A is—an untenable proposition. Thus set theorists have adopted an axiom (of *foundation*) which precludes bottomless (unfounded) chains such as

$$\ldots \in d \in c \in b \in a$$

The impossibility of $A \in A$ follows from this, since otherwise it would lead to $\ldots \in A \in A \in A \in A$.

[40] All reasonable axioms are based on intuitively acceptable concepts. The idea that sets are formed by stages led to many nice axioms of axiomatic set theory.

Ordered triples, quadruples and beyond can be easily defined using the ordered pair as the basic building block:

1.1.2.25 Definition. (Ordered Tuples) We define the symbol $\langle x_1, \ldots, x_n \rangle$, pronounced the *ordered n-tuple* or just—n-tuple—by two recurrence equations:

$$\begin{aligned} \langle x_1, x_2 \rangle &= (x_1, x_2) \\ &\text{and, for } n \geq 2, \\ \langle x_1, \ldots, x_n, x_{n+1} \rangle &= (\langle x_1, \ldots, x_n \rangle, x_{n+1}) \end{aligned}$$

x_i is the i-th component of the tuple. $\langle a, b \rangle$ is also called an *2-tuple* (as well as an ordered pair).

We often employ the abbreviation $\vec{x}_n$ for the (ordered) sequence $x_1, \ldots, x_n$. The presence of "$\vec{}$" will not permit the confusion between the sequence $\vec{x}_n$ and the component x_n. If the length n is immaterial or known, we may just write $\vec{x}$. □

The above is a simple *recursive* or inductive *definition*. It compactifies and renders finite an *infinite-length definition* such as:

$$\begin{aligned} \langle x, y \rangle &= (y, y) \\ \langle x, y, z \rangle &= (\langle x, y \rangle, z) \\ \langle x, y, z, u \rangle &= (\langle x, y, z \rangle, u) \\ \langle x, y, z, u, w \rangle &= (\langle x, y, z, u \rangle, w) \\ &\vdots \end{aligned}$$

In essence, it *finitely describes* the "$\vdots$" above.

This is entirely analogous with *loops* in programming where a variable-length (and therefore syntactically illegal)—it depends on the value of N—program segment is correctly implemented as a loop; that is, the following, where $X \leftarrow X + 1$ occurs N times

read N, X
$X \leftarrow 0$
$X \leftarrow X + 1$
$\vdots$
$X \leftarrow X + 1$

is captured by this

read N, X
$X \leftarrow 0$
repeat N **times**
{
$X \leftarrow X + 1$

}

The reader has seen recursive definitions similar to the one in 1.1.2.25, for example, the one that defines nonnegative (integer) powers of a non zero real number a by two equations:

$a^0 = 1$
and, for $n \geq 0$,
$a^{n+1} = a \cdot a^n$

Recursive definitions of this and of more general types are reviewed in Section 1.4.

1.1.2.26 Exercise. Show that the name ordered (4-) tuple is apt for $\langle x, y, z, w \rangle$ by showing that $\langle x, y, z, w \rangle = \langle X, Y, Z, W \rangle$ implies that $x = X$, $y = Y$, $z = Z$ and $w = W$. □

1.1.2.27 Exercise. Write down explicitly the set for which the tuple $\langle x, y, z, w \rangle$ is *compact notation*. □

1.1.2.28 Definition. (Cartesian Product) Let $A_1, \ldots, A_n$ be sets. Then their *Cartesian product, in the given order*, is the set

$$\Big\{ \langle a_1, \ldots, a_n \rangle : a_i \in A_i, \text{for } i = 1, \ldots, n \Big\}$$

We will employ the symbols

$$\bigtimes_{1 \leq i \leq n} A_i \quad \text{or} \quad \bigtimes_{i=1}^{n} A_i$$

as alternative shorthands for this product.

If $A_1 = A$ and $A_2 = B$ then we write $A \times B$ rather than $\bigtimes_{i=1}^{2} A_i$. It is all right, but sloppy, to write $A_1 \times \cdots \times A_n$ for the general case. If $A_i = C$, for all i, then we write C^n for $\bigtimes_{i=1}^{n} A_i$. □

1.1.2.29 Example. $\{1\} \times \{2\} = \{\langle 1, 2 \rangle\}$ and $\{2\} \times \{1\} = \{\langle 2, 1 \rangle\}$. Thus $\{1\} \times \{2\} \neq \{2\} \times \{1\}$; the Cartesian product is not commutative in general. □

$\bigtimes_{i=1}^{n} A_i$ can be given by a simple recursive (inductive) definition:

$\bigtimes_{i=1}^{1} A_i = A_1$
and, for $n \geq 1$,
$\bigtimes_{i=1}^{n+1} A_i = \left(\bigtimes_{i=1}^{n} A_i \right) \times A_{n+1}$

The reader should verify that this is consistent with 1.1.2.28 and 1.1.2.25.

Similarly, A^n can be defined inductively (recursively) as

$A^1 = A$

and, for $n \geq 1$,

$A^{n+1} = A^n \times A$

1.1.2.30 Example. $A \times \emptyset = \emptyset$, since $\langle x, y \rangle \in A \times \emptyset$ is equivalent to $x \in A \wedge y \in \emptyset$, which is false. Similarly, $\emptyset \times A = \emptyset$. □

We conclude our review of set operations with the *power set*.

1.1.2.31 Definition. (Power Set) For any set A, its *power set*—denoted as $\mathcal{P}(A)$ or 2^A—is $\{x : x \subseteq A\}$. □

1.1.2.32 Example. Thus, $2^\emptyset = \{\emptyset\}$; $2^{\{\emptyset\}} = \{\emptyset, \{\emptyset\}\}$; and $2^{\{\emptyset,\{\emptyset\}\}} = \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \{\emptyset, \{\emptyset\}\}\}$.

$2^{\{0,1\}} = \{\emptyset, \{0\}, \{1\}, \{0,1\}\}$.

Since $\emptyset \subseteq A$ and $A \subseteq A$ for any set A, we have always $\emptyset \in 2^A$ and $A \in 2^A$. □

1.1.3 Alphabets, Strings and Languages

A *string* or *expression* or a *word* is just a tuple, all of whose components come from the same set, A, the latter being called the *alphabet*. We say that "*x is a string of length n* over *the alphabet A*" meaning $x \in A^n$.

Traditionally, strings are written down without separating commas or spaces, nor with enclosing angular brackets. So if $A = \{a, b\}$ we will write $aababa$ rather than $\langle a, a, b, a, b, a \rangle$.

Concatenation of $\langle a_1, \ldots, a_m \rangle$ and $\langle b_1, \ldots, b_n \rangle$ in that order, denoted as

$$\langle a_1, \ldots, a_m \rangle * \langle b_1, \ldots, b_n \rangle$$

is the string of length $m + n$

$$\langle a_1, \ldots, a_m, b_1, \ldots, b_n \rangle$$

Clearly, concatenation as defined above is *associative*, that is, for any strings x, y and z we have $(x * y) * z = x * (y * z)$.

It is convenient to introduce a *null* or *empty* string, that has no members, and hence has length 0. We will denote it by ϵ. We will not attempt to give it a precise tuple counterpart, but some people write "$\langle\rangle$" with nothing between brackets.

At the intuitive level, and given how concatenation was defined, we see that $x * \epsilon = \epsilon * x = x$ for any string x. We will distinguish $\emptyset$ and ϵ since one is an "unordered set" will the other is ordered; but both are empty.

The set of *all strings of non zero length* over A is denoted by A^+. This is, of course,

$$\bigcup_{i=1}^{\infty} A^i$$

Adding ϵ to the above we get the unqualified set of all strings over A, denoted by A^*; that is, $A^* = A^+ \cup \{\epsilon\}$. This set is often called *the Kleene star* of A.

A string A is a *prefix* of a string B if there is a string C such that $B = A * C$. It is a *suffix* of B if for some D, we have $B = D * A$. The prefix (suffix) is *proper* if it is not equal to B.

Just as we use *implied multiplication*, ab for $a \times b$ or $a \cdot b$, we also use *implied concatenation*, xy for $x * y$ —leaving it up to the context to fend off ambiguities.

1.1.3.1 Example. Not all alphabets are amenable to writing tuples in "string-notation". For example, $A = \{1, 11\}$ has a problem. The notation 111 is ambiguous: Do we mean $\langle 1, 1, 1\rangle$, $\langle 11, 1\rangle$, or $\langle 1, 11\rangle$? □

1.1.3.2 Definition. (Languages) A *language*, L, over an alphabet A is just a subset of A^*. □

The "interesting" languages are those that are *finitely definable*. *Automata and language theory* studies the properties of such finitely definable languages and of the "machinery" that effects these finite definitions.

1.1.3.3 Definition. (Concatenation of Languages) If L and M are two languages over an alphabet A, then the symbol $L * M$ or simply (implied concatenation) LM means the set $\{xy : x \in L \wedge y \in M\}$. □

One can learn to live with $*$ as both a unary (one-argument) operation, A^*, and as a binary one, $L * M$, much the same way we can see no ambiguity in uses of minus as $-x$ and $y - z$.

1.2 RELATIONS AND FUNCTIONS

Intuitively, a relation is a formula, $\mathscr{A}(x, y, z)$. We say that a, b, c are related according to $\mathscr{A}(x, y, z)$ just in case $\mathscr{A}(a, b, c)$ is true. Influenced by the set theorist who wants to realize "everything" (even formulae) as some set, the modern mathematician views relations *extensionally* (by what they contain) *as sets*. For example, $\mathscr{A}(x, y, z)$ naturally defines this set, its *extension*: $\{\langle x, y, z\rangle : \mathscr{A}(x, y, z)\}$. One goes one step further and forgets the role of $\mathscr{A}$. As a result, we give a totally extensional definition of a relation as a set of tuples, disregarding how it may have been formed by a "defining property".

1.2.0.4 Definition. A *binary relation* —or simply *relation*—R is a set of 2-tuples. We use the notations $\langle x, y\rangle \in R$, xRy and $R(x, y)$ to *mean the same thing*. □

A relation R, on the other hand, immediately gives rise to an atomic *formula* — variably denoted by one of the forms $\langle x, y\rangle \in R$, xRy or $R(x, y)$— just as the specific relations $<$ and $\in$ lead to the atomic formulae $x < y$ and $x \in y$ (cf. 1.1.1.1).

Pause. So, every formula $\mathscr{A}(x, \ldots)$ defines the relation $A = \{\langle x, \ldots\rangle : \mathscr{A}(x, \ldots)\}$, and every (binary) relation R defines the (atomic) formula xRy; right?◀

Not exactly. If we have an "enormous"[41] supply of symbols for formulae, then we could do this, since for every relation we could then introduce a formula symbol (so-called "predicate")—say, R, ϕ, $\prec$, or whatever—by a *definition*, such as "$x \prec y$ if and only iff $\langle x, y\rangle$ is a member of the given relation". This fails in most practical cases, e.g., in set theory and arithmetic, where our symbol-alphabet is finite or enumerable (cf. Definition 1.3.0.40). To write down —that is, to "have"—a formula, we need *notation* for it. As we will see in Section 1.3, we have far "more" binary relations R than we have means to "write them down" as formulae xRy, *if our symbol-alphabet is finite or enumerable.*

Hm. Did I not just write down "xRy"? Well, yes; however, writing one or two symbols down, like "R" or "Q" and saying that they "stand for relations" does not equate to having *a system of notation to write down* all *binary relations.*

Intuitively, a relation is a table—possibly infinite in length—of pairs like

x	z
a_1^1	a_2^1
a_1^2	a_2^2
$\vdots$	$\vdots$

The head-row names the relation's variables. The entries in each row represent the tuples-members of the relation. It is standard convention to think of the left column, headed by x as the "input-side", while the right column as the "output-side". This is consistent with a "black box" view of the relation

$$a_1^i \longrightarrow \boxed{} \longrightarrow a_2^i$$

where we don't know or don't care what makes it tick, but we do know which inputs cause which output(s).

It is not *a priori* precluded to have the same input produce several outputs. For example, think of $R = \{\langle 1, 2\rangle, \langle 1, 1\rangle, \langle 1, 7\rangle\}$.

Thus, the relation (table) establishes a *one-to-many* input/output correspondence.

Contrary to our viewpoint with formulae $\mathscr{A}(x, y)$—where the *input* variables *are* all the free variables, here x and y—in the case of relations we are allowed *two points of view*, one being the one presented above, and the other where *both* x and y are the inputs of the relation $R(x, y)$. The context will fend for us!

[41] See also Section 1.3 to appreciate that not all infinities are equal in size.

Of course, when we take *all* the variables of a relation as input, then the output that is implied—just as in the case of formulae—is one of **t** or **f**.

Since $\langle \vec{a}_{n+1} \rangle = (\langle \vec{a}_n \rangle, a_{n+1})$, *there is no loss of generality in focusing mostly on binary relations. In other words, the left (input) column may well be a column of* n*-tuple entries* $a_1^j = \langle A_1^j, A_2^j, \ldots, A_n^j \rangle$. *The relation is then said to be* $(n+1)$*-ary and, in table form, would look like*

x_1	...	x_n	z
A_1^1	...	A_n^1	a_1^1
A_1^2	...	A_n^2	a_2^2
$\vdots$		$\vdots$	$\vdots$

The set consisting of the entries in the input column is the relation's domain—that is, those inputs that cause some output—while those in the output column constitute the range—that is, the set of all outputs.

1.2.0.5 Definition. Let R be a (binary) relation. Its *domain*, denoted by $\text{dom}(R)$, is the set $\{x : (\exists y) xRy\}$. Its *range*, denoted by $\text{ran}(R)$, is the set $\{x : (\exists y) yRx\}$. □

1.2.0.6 Example. Let $R = \{\langle x, x \rangle : x \in \mathbb{N}\}$. Then $\text{dom}(R) = \text{ran}(R) = \mathbb{N}$.
Let $Q = \{\langle 0, x \rangle : x \in \mathbb{N}\}$. Then $\text{dom}(Q) = \{0\}$ and $\text{ran}(Q) = \mathbb{N}$.
Let $S = \{\langle x, 0 \rangle : x \in \mathbb{N}\}$. Then $\text{ran}(S) = \{0\}$ and $\text{dom}(S) = \mathbb{N}$.
Let $T = \{\langle 0, 0 \rangle, \langle 0, 7 \rangle\}$. Then $\text{dom}(T) = \{0\}$ and $\text{ran}(T) = \{0, 7\}$. □

An abstract *term* of logic captures well the intentional aspect of a *function*—indeed a *function call*—of mathematics and programming: we have a "rule" that defines the input/output dependence. For example, the "rule" $x + y$ that tells us how the output is to be obtained, once we have the x and y values.

While, in logic, a term is a totally different type of object from a formula, on the other hand, extensionally—i.e., in its set theory realization—a function is a subsidiary construct of a relation. Referring back to the black box analogy, a function is simply a relation that obeys the restriction that *no input can cause more than one output*. So a function, extensionally, is a *single-valued* relation.

1.2.0.7 Definition. A *function* R is a *single-valued* relation. That is, whenever we have both xRy and xRz, we will also have $y = z$.

It is traditional to use, generically, lower case letters from among f, g, h, k to denote functions but this is by no means a requirement. □

1.2.0.8 Example. The empty set is a relation of course, the empty set of pairs. It is also a function since

$$\langle x, y \rangle \in \emptyset \wedge \langle x, z \rangle \in \emptyset \rightarrow y = z$$

vacuously, by virtue of the left hand side of $\rightarrow$ being false. □

It is often the case that we study relations, and functions, that take their inputs from a given set A that is fixed throughout the study, and, similarly, produce their outputs in a given fixed set B.

For example, our work on computability in this volume deals exclusively with functions and relations whose inputs and outputs are from $\mathbb{N}$.

Additional terminology has been invented to name these fixed "input-" and "output-spaces" and also to name relations that *fully* utilize one or the other of these spaces. The input space is called the *left field* while the output space is called the *right field*.

If A and B are the adopted left and right fields of the function or relation R then clearly $R \subseteq A \times B$, and, in particular, $\text{dom}(R) \subseteq A$ while $\text{ran}(R) \subseteq B$. A well-established abbreviation—other than $R \subseteq A \times B$— for "R is relation with left field A and right field B" is $R : A \to B$, read "R is a relation from A to B".

If $A = B$, then we say "R is a relation *on* A".

If $\text{dom}(R) = A$, then R is *totally defined* on A. We just say "R is *total*". If $\text{ran}(R) = B$, then R "covers" the entire right field with its outputs. We say "R is *onto*".

Pause. Totalness and ontoness are *relative* to a left field and a right field, respectively; they are *not absolute notions.*◀

A relation $R : A \to B$ is either total or not (*nontotal*). An indifference toward which is which will be expressed by calling R *partial*. Thus "partial" is *not* synonymous with "nontotal". *All* relations are therefore *partial relations.*

All the terminology introduced in this  *-segment applies to the special case of functions as well.* 

We now turn to notation and concepts specific to functions. Let f be a function. First off, $f(a)$ denotes the unique b such that afb or $\langle a, b\rangle \in f$. Note that such a b exists iff $a \in \text{dom}(f)$. Thus

$$b = f(a) \text{ iff } \langle a, b\rangle \in f \text{ iff } afb$$

We write $f(a)\downarrow$—pronounced "$f(a)$ is defined" or "$f(a)$ *converges*"—to mean $a \in \text{dom}(f)$. Otherwise we write $f(a)\uparrow$—pronounced "$f(a)$ is undefined" or "$f(a)$ *diverges*".

The set of *all* outputs of a function, when the inputs come from a particular set X, is called the *image of* X *under* f and is denoted by $f_{\rightarrow}(X)$. Thus,

$$f_{\rightarrow}(X) = \{f(x) : x \in X\} \tag{1}$$

Pause. So far we have been giving definitions regarding functions of one variable. Or have we?◀

Not really: We have already said that the multiple-input case is subsumed by our notation. If $f : A \to B$ and A is a set of n-tuples, then f is a function of "n-variables", essentially. We usually abuse the notation $f(\langle \vec{x}_n \rangle)$ and write instead $f(\vec{x}_n)$.

The *inverse image* of a set Y under a function is useful as well, that is, the set of *all* inputs the generate f-outputs in Y. It is denoted by $f_{\leftarrow}(Y)$ and is defined as

$$f_{\leftarrow}(Y) = \{x : f(x) \in Y\} \tag{2}$$

Regarding, say, the definition of $f_{\rightarrow}$:

What if $f(a)\uparrow$? How do you "collect" an undefined value *into a set?*

Well, you don't. Both (1) and (2) have a rendering that is independent of the notation "$f(a)$".

$$f_{\rightarrow}(X) = \{y : (\exists x \in X)\langle x, y\rangle \in f\} \tag{1'}$$

$$f_{\leftarrow}(Y) = \{x : (\exists y \in Y)\langle x, y\rangle \in f\} \tag{2'}$$

1.2.0.9 Example. Thus, $f_{\rightarrow}(\{a\}) = \{f(x) : x \in \{a\}\} = \{f(x) : x = a\} = \{f(a)\}$.

Let now $g = \{\langle 1, 2\rangle, \langle\{1, 2\}, 2\rangle, \langle 2, 7\rangle\}$. Thus, $g(\{1, 2\}) = 2$, but $g_{\rightarrow}(\{1, 2\}) = \{2, 7\}$. Also, $g(5)\uparrow$ and $g_{\rightarrow}(\{5\}) = \emptyset$.

On the other hand, $g_{\leftarrow}(\{2, 7\}) = \{1, \{1, 2\}, 2\}$ and $g_{\leftarrow}(\{2\}) = \{1, \{1, 2\}\}$, while $g_{\leftarrow}(\{8\}) = \emptyset$. □

When $f(a)\downarrow$, then $f(a) = f(a)$ as is naturally expected. What about when $f(a)\uparrow$? This begs a more general question that we settle as follows:

First, seeking help from logic. For any formula $\mathscr{A}[x]$ and term t *that does not contain the variable* x,

$$\vdash \mathscr{A}[t] \equiv (\exists x)(x = t \wedge \mathscr{A}[x]) \tag{1}$$

We settle (1) by a ping pong argument (putting aside an urge to proclaim "but, it is obvious!").

($\rightarrow$) direction. We want to prove $\mathscr{A}[t] \rightarrow (\exists x)(x = t \wedge \mathscr{A}[x])$. Note that

$$\mathscr{A}[t] \rightarrow t = t \wedge \mathscr{A}[t]$$

is true. So is

$$t = t \wedge \mathscr{A}[t] \rightarrow (\exists x)(x = t \wedge \mathscr{A}[x])$$

by 1.1.1.40 since we may view $x = t \wedge \mathscr{A}[x]$ as $(x = t \wedge \mathscr{A}[x])[x]$ and thus view $t = t \wedge \mathscr{A}[t]$ as $(x = t \wedge \mathscr{A}[x])[t]$ due to the absence of x in t. Using this and the previous displayed formula along with tautological implication we get what we want.

($\leftarrow$) direction. We want to prove $(\exists x)(x = t \wedge \mathscr{A}[x]) \rightarrow \mathscr{A}[t]$. We will employ the deduction theorem, so we *freeze all free variables* in $(\exists x)(x = t \wedge \mathscr{A}[x])$, and *assume* it. So, let us call a an x-value that makes the quantification work (cf. 1.1.1.13). We have

$$a = t \wedge \mathscr{A}[a] \tag{2}$$

Since the $a = t$ part of (2) and the Leibniz axiom [(vi) of 1.1.1.38] yield $\mathscr{A}[a] \equiv \mathscr{A}[t]$, the remaining part of (2) yields the truth of $\mathscr{A}[t]$, as needed.

Transferring the above result to the specific case of substituting terms into input variables[42] of *relations*, we have the following.

1.2.0.10 Remark. For any $(m+n+1$-ary) relation $R(z_1,\ldots,z_m,x,y_1,\ldots,y_n)$, function f, and object a, the substitution $R(z_1,\ldots,z_m,f(a),y_1,\ldots,y_n)$ is shorthand for

$$(\exists w)\Big(w = f(a) \wedge R(z_1,\ldots,z_m,w,y_1,\ldots,y_n)\Big) \tag{3}$$

Note that $w = f(a)$ entails that $f(a)\downarrow$, so that if *no such* w *exists* [the case where $f(a)\uparrow$], then (3) is *false*; *not* undefined!

This convention is prevalent in the modern literature [cf. Hinman (1978), p. 9]. Contrast with the convention in Kleene (1943), where, for example, an expression like $f(a) = g(b)$ [and even $f(a) = b$] is allowed to be undefined! □

1.2.0.11 Example. Thus, applying the above twice, $f(a) = g(b)$ means $(\exists u)(\exists w)(u = f(a) \wedge w = g(b))$ which simplifies to $(\exists u)(u = f(a) \wedge u = g(b))$. In particular, $f(a) = g(b)$ entails that $f(a)\downarrow$ and $g(b)\downarrow$. □

The above is unsettling as it fails to satisfy the reflexivity of equality [axiom (v) of 1.1.1.38]: If $f(a)\uparrow$, then $\neg f(a) = f(a)$. To get around this difficulty, Kleene (1943) has *extended* equality to include the *undefined* case, restoring reflexivity in this "generalized" equality relation. We will use this so-called *Kleene-complete-equality* quite often in the chapter on computability. This version of equality uses a different symbol, $\simeq$, to avoid confusion with the "standard" equality, =, of Remark 1.2.0.10 that compares only objects (not "undefined values"). For any two functions f and g, we define

$$f(a) \simeq g(b) \stackrel{\text{Def}}{\equiv} f(a)\uparrow \wedge g(b)\uparrow \vee (f(a)\downarrow \wedge g(b)\downarrow \wedge f(a) = g(b))$$

while $f(a) \simeq b$ means the same thing as afb, that is, $f(a) = b$.

1.2.0.12 Example. Let $g = \{\langle 1,2\rangle, \langle\{1,2\},2\rangle, \langle 2,7\rangle\}$. Then, $g(1) = g(\{1,2\})$ and also $g(1) \simeq g(\{1,2\})$. Also, $g(1) \not\simeq g(2)$ and also $g(1) \neq g(2)$. Moreover, $g(3) \simeq g(9)$. □

If f and g are functions and $f \subseteq g$ then g is an *extension* of f while f is a *restriction* of g. If $g : A \to B$, one way to restrict g to f is to choose for f a "smaller" left field, $C \subseteq A$, and take for f only those 2-tuples that have the their first component in C. We write this as $f = g \restriction C$. Thus, $g \restriction C = g \cap (C \times B)$.

Note that every function f extends the *totally undefined function* $\emptyset$ since $\emptyset \subseteq f$.

1.2.0.13 Definition. A function f is 1-1 if for all x and y, $f(x) = f(y)$ implies $x = y$. □

Note that $f(x) = f(y)$ implies that $f(x)\downarrow$ and $f(y)\downarrow$ (1.2.0.10).

[42] Here we view *every* variable of R as input; output is **t** or **f**. Cf. discussion on p. 42.

1.2.0.14 Example. $\{\langle 1,1\rangle\}$ and $\{\langle 1,1\rangle, \langle 2,7\rangle\}$ are 1-1. $\{\langle 1,0\rangle, \langle 2,0\rangle\}$ is not. $\emptyset$ is 1-1 vacuously. □

1.2.0.15 Definition. (Relational Converse) If R is a relation, then its *converse*, denoted by R^{-1} is the relation $\{\langle x,y\rangle : yRx\}$. □

1.2.0.16 Exercise. Prove that if f is a 1-1 function, then the relation converse f^{-1} is a function (that is, single-valued). □

1.2.0.17 Definition. (1-1 Correspondence) A function $f : A \to B$ is called a *1-1 correspondence* iff it is all three: 1-1, total and onto.

Often we say that A and B are *in 1-1 correspondence* writing $A \sim B$, omitting mention of the function that *is* the 1-1 correspondence. □

The terminology is derived from the fact that every element of A is paired with precisely one element of B and vice versa.

1.2.0.18 Definition. (Composition of Relations and Functions) Let $R : A \to B$ and $Q : B \to C$ be two relations. The relation $R \circ Q : A \to C$, their *relational composition*, is the relation

$$\Big\{\langle x,y\rangle : (\exists z)(xRz \wedge zQz)\Big\} \tag{1}$$

If R and Q are functions, then their *functional composition*—or composition as functions—refers to their relational composition, but has a different notation: (QR) (no "$\circ$") is an alternative notation for $R \circ Q$; *note the order reversal.* □

So $xR \circ Qy$ iff $(\exists z)(xRz \wedge zQy)$. Let then $xR \circ Qy$ and also $xR \circ Qw$. For some a and b, guaranteed to exist, we have xRa and aQy on one hand and xRb and bQw on the other. Let next R and Q both be functions. Then $a = b$ (from R) and hence $y = w$ (from Q). Thus,

If R and Q are functions, then so is their composition, $R \circ Q$ or (QR).

Let R and Q still be functions. Assume that $(QR)(a) \downarrow$. Then, for some b, $aR \circ Qb$, and hence, for some c, aRc and cQb. That is,

$$R(a) = c \text{ and } Q(c) = b. \text{ For short, } (QR)(a) = Q(R(a)) \tag{2}$$

The above justifies the order reversal for the alternative notation of "functional composition".

1.2.0.19 Theorem. *Relational composition is associative, that is, $R \circ (Q \circ S) = (R \circ Q) \circ S$ for any relations R, Q, S. If the relations are functions we may also write $((SQ)R) = (S(QR))$.*

Proof. See Exercise 24 in Section 1.8. □

1.2.0.20 Definition. The *identity function* on a set A is $\mathbf{1}_A : A \to A$ given by $\mathbf{1}_A(x) = x$ for all $x \in A$. □

By 1.2.0.19, if R, Q, T, S are relations, then $R \circ Q \circ T \circ S$ is *unambiguous*, as it means the same thing regardless of how we insert brackets. In particular,

$$\underbrace{R \circ R \circ \cdots \circ R}_{n \geq 1 \text{ copies of } R}$$

is unambiguous regardless of the absence of brackets. We have the shorthand R^n for the above chain of compositions. We can put this into an inductive definition similar to the one that defines positive powers of a positive real:

1.2.0.21 Definition. (Relational Powers) The symbol R^n, for $n \geq 1$, is the *relational power* of R and is defined as

$$R^1 = R$$

and, for $n \geq 1$,

$$R^{n+1} = R \circ R^n$$

If R is a relation on A, then we replace the first equation by $R^0 = \mathbf{1}_A$ and the condition for the second becomes "and, for $n \geq 0$". □

The following interesting result connects the notions of ontoness and 1-1ness with the "algebra" of composition.

1.2.0.22 Theorem. *Let $f : A \to B$ and $g : B \to A$ be functions. If $(gf) = \mathbf{1}_A$, then g is onto while f is total* and *1-1.*

We say that g is a *left inverse* of f and f is a *right inverse* of g.

Proof. **About g:** Our goal, ontoness, means that, for each $x \in A$, a y exists such that $g(y) = x$. Fix then an $x \in A$. By $(gf) = \mathbf{1}_A$, we have $(gf)(x) = \mathbf{1}_A(x) = x$. But $(gf)(x) = g(f(x))$. So take $y = f(x)$.

About f: As seen above, $x = g(f(x))$ for each $x \in A$. Since this is the same as "$xf \circ gx$ is true", there must be a z such that xfz and zgx. The first of these says $f(x) = z$ and therefore $f(x) \downarrow$. This settles totalness.

For the 1-1ness, let $f(a) = f(b)$. Applying g to both sides (that is, using Exercise 1.8.4) we get $g(f(a)) = g(f(b))$. But this says $a = b$, by $(gf) = \mathbf{1}_A$, and we are done. □

1.2.0.23 Example. *The above is as much as we can be expected to prove.* For example, say $A = \{1, 2\}$ and $B = \{3, 4, 5, 6\}$. Let $f : A \to B$ be $\{\langle 1, 4\rangle, \langle 2, 3\rangle\}$ and $g : B \to A$ be $\{\langle 4, 1\rangle, \langle 3, 2\rangle, \langle 6, 1\rangle\}$, or in friendlier notation

$f(1) = 4$
$f(2) = 3$

and
$g(3) = 2$
$g(4) = 1$
$g(5)\uparrow$
$g(6) = 1$

Clearly, $(gf) = \mathbf{1}_A$ holds, but note:
(1) f is not onto.
(2) g is neither 1-1 nor total. □

1.2.0.24 Example. With $A = \{1, 2\}$, $B = \{3, 4, 5, 6\}$ and $f : A \to B$ and $g : B \to A$ as in the previous example, consider also the functions $\tilde{f}$ and $\tilde{g}$ given by

$\tilde{f}(1) = 6$
$\tilde{f}(2) = 3$
and
$\tilde{g}(3) = 2$
$\tilde{g}(4) = 1$
$\tilde{g}(5)\uparrow$
$\tilde{g}(6) = 2$

Clearly, $(\tilde{g}f) = \mathbf{1}_A$ and $(g\tilde{f}) = \mathbf{1}_A$ hold, but note:
(1) $f \neq \tilde{f}$.
(2) $g \neq \tilde{g}$.
Thus, neither left nor right inverses need to be unique. The article "a" in the definition of said inverses was well-chosen. □

The following two partial converses of 1.2.0.22 are useful.

1.2.0.25 Theorem. *Let $f : A \to B$ be total and 1-1. Then there is an* onto $g : B \to A$ *such that $(gf) = \mathbf{1}_A$.*

Proof. Consider the converse relation (1.2.0.15) of f and call it g:

$$g \stackrel{\text{Def}}{=} \{\langle x, y\rangle : f(y) = x\} \tag{1}$$

By Exercise 1.2.0.16, $g : B \to A$ is a (possibly nontotal) function. Note that, for any $a \in A$, there is a b such that $f(a) = b$ (f is total), and, by (1), $g(b) = a$. That is, $g(f(a)) = a$, or $(gf) = \mathbf{1}_A$. □

1.2.0.26 Remark. By (1) above, $\text{dom}(g) = \{x : (\exists y)\, \langle x, y\rangle \in g\} = \{x : (\exists y) f(y) = x\} = \text{ran}(f)$. □

1.2.0.27 Theorem. *Let $f : A \to B$ be onto. Then there is a* total and 1-1 $g : B \to A$ *such that $(fg) = \mathbf{1}_B$.*

Proof. By assumption, $\emptyset \neq f_{\leftarrow}(\{b\}) \subseteq A$, for all $b \in B$. To define $g(b)$ choose *one* $c \in f_{\leftarrow}(\{b\})$ and set $g(b) = c$. Since $f(c) = b$, we get $f(g(b)) = b$ for all $b \in B$, and hence g is 1-1 and onto by 1.2.0.22. □

The above argument makes potentially infinitely many choices, one from each $f_{\leftarrow}(\{b\})$. Of course, these sets are pairwise disjoint.

Pause. Why is it that $f_{\leftarrow}(\{x\}) \cap f_{\leftarrow}(\{y\}) = \emptyset$ if $x \neq y$?◀

Contrast with the case where $B = \{b, b'\}$, a set of two elements. Then we can define g by simply saying

Let $c \in f_{\leftarrow}(\{b\})$, *and set* $g(b) = c$. *Let* $c' \in f_{\leftarrow}(\{b'\})$, *and set* $g(b') = c'$.

We can contain our (two) choices in the space of a proof. The same is true if B had 2^{350000} elements. We would just have to write a proof that would be, well, a bit longer, using a copy of the sentence "Let $y \in f_{\leftarrow}(\{x\})$, and set $g(x) = y$" once for each one of the 2^{350000} members of B that we generically called here "x".

However, the "Let . . . " approach does not work for an infinite B, since we cannot contain infinitely many such sentences in the space of a finite-length proof; *unless* we have a way to *codify the infinitely many choices in a finite manner*. For example, if A is a set of natural numbers then so is $f_{\leftarrow}(\{b\})$ for each b and we can say precisely how a c can be chosen in each case: For example, "for each $b \in B$, choose the smallest c in $f_{\leftarrow}(\{b\})$" would do just fine.

Some mathematicians did not accept that one may effect infinitely many choices, *in the absence of* a finitely describable process of how to go about making them; this was not mathematically acceptable. They argued that in the absence of some kind of known "structure" in the various $f_{\leftarrow}(\{b\})$, all the elements of these sets "look the same" and therefore the infinite process of "choosing" cannot be compacted into a finite *well-defined* description. This observation hinges on the number of choices one needs to make rather than on the number of elements in a $f_{\leftarrow}(\{b\})$.

An example of the difficulty, in layman's terms, attributed to Russell, contrasts two cases: One where we have an infinite set of pairs of shoes, and another, where we have an infinite set of pairs of socks.

In the former case we can finitely define infinitely many choices of one shoe per pair by always choosing the *left* shoe in each pair. In the case of socks this "rule" does not define *well* which sock to pick, because, the two socks in a pair have no distinct "left" or "right" members.

I used past tense above, "Some mathematicians did not accept, etc.", for the dissenting opinion. This is because mathematicians nowadays feel comfortable with the notion of effecting infinitely many choices *without having a finite process to describe said choices*. They even have an axiom (the Axiom of Choice, or AC) that says they can do so [for a thorough discussion of AC, see Tourlakis (2003b)].

1.2.0.28 Definition. (Equivalence Relations) Let R be a relation on a set A. We call it an *equivalence relation* iff it has all the three following properties:

(1) It is *reflexive*, that is, xRx holds, for all $x \in A$

(2) It is *symmetric*, that is, xRy implies yRx, for all x and y

(3) It is *transitive*, that is, xRy and yRz imply xRz, for all x, y and z. □

The concept "equivalence relation" does not apply to relations $R : A \to B$ with $A \neq B$. The concept of reflexivity requires reference to the left (and right, since they are equal) field. If we make the fields larger, without adding any pairs to the relation, a previously reflexive relation will cease being reflexive.

1.2.0.29 Example. The function $\mathbf{1}_A : A \to A$ is an equivalence relation.

The relation $<$ on $\mathbb{N}$ is transitive, but neither symmetric, nor reflexive; on the other hand, $\leq$ has reflexivity (still fails symmetry).

The relation R on $\mathbb{Z}$ given by: "xRy iff the difference $x - y$ is divisible by 5" (divisible with 0 remainder, that is) can be easily verified to be an equivalence relation. □

Given an equivalence relation R on a set A, we define for each $x \in A$ the set of all its equivalents in A. This is known as an *equivalence class* of R. We employ the symbol $[x]_R$, thus

$$[x]_R \stackrel{\text{Def}}{=} \{y \in A : xRy\} \tag{1}$$

Despite employing the term "class" in this context, which is standard practice in the literature, we do not imply at all that these classes are "too large" to technically be sets. On the contrary, any such class is a subset of A.

1.2.0.30 Theorem. *Given an equivalence relation R on A. Its equivalence classes $[x]_R$ satisfy*

(1) $[x]_R \neq \emptyset$

(2) *if xRy iff* $[x]_R = [y]_R$

(3) *if* $[x]_R \cap [y]_R \neq \emptyset$, *then* $[x]_R = [y]_R$

(4) $\bigcup_{x \in A} [x]_R = A$

Proof.

(1) $[x]_R \neq \emptyset$: In fact $x \in [x]_R$ by xRx.

(2) if xRy iff $[x]_R = [y]_R$:

First, *assume the left hand side of the "iff"*, which also yields yRx by symmetry. For the $(\subseteq)$ of the right hand side let $z \in [x]_R$. Thus xRz. Transitivity yields yRz, hence $z \in [y]_R$.

For the $(\supseteq)$, let $z \in [y]_R$, i.e., yRz. Along with xRy and transitivity we have xRz, that is, $z \in [x]_R$.

Now *assume the right hand side of the "iff"*. By the proof of (1), $y \in [x]_R$, thus xRy.

(3) if $[x]_R \cap [y]_R \neq \emptyset$, then $[x]_R = [y]_R$: By the assumption, there is a z such that $z \in [x]_R$ and $z \in [y]_R$. Thus xRz and yRz, the latter implying zRy. By transitivity, xRy; done by (2).

(4) $\bigcup_{x\in A}[x]_R = A$: The ($\subseteq$) is trivial, since $[x]_R \subseteq A$ for any $x \in A$. For ($\supseteq$), let $x \in A$. But $x \in [x]_R$ and this meets the entrance requirement for x in $\bigcup_{x\in A}[x]_R$. □

1.2.0.31 Remark. (Partitions) Thus the equivalence classes of an R on A meet the three conditions a *partition* of A must satisfy, by definition:

A *partition* on A is *a set of sets* P such that

(a) If $C \in P$, then $C \neq \emptyset$

(b) (Nonoverlap) If C and D are in P and $C \cap D \neq \emptyset$, then $C = D$

(c) (Coverage) $\bigcup S = A$.

So equivalence classes furnish an example of partitions. More is true (cf. Exercise 1.8.33): If P is a partition on A, then an equivalence relation on A can be defined in a natural way, whose equivalence classes are precisely the members of P. □

1.2.0.32 Definition. (Order) A relation R on a set A is called an *order* or *order relation* iff it is transitive and *irreflexive*, the latter meaning $\neg(\exists x)xRx$.

We normally use the abstract symbol $<$ for orders and let the context fend off confusion with concrete usage of the symbol as the order on $\mathbb{N}$ or $\mathbb{R}$. □

We call all orders *partial*, since some orders, $<$ on A, are *total* or *linear*, while others are not.

Indeed, we will seldom use the qualifier "partial" for orders as it is automatically understood. Exception*: Often one presents the "package" consisting of the order* and *the underlying set A together, in symbols* $(A, <)$*, and calls it a* partially ordered set *or* POset.

That an order $<$ on A is total means that every pair of members x and y of A are *comparable*: That is, one of $x = y$, $x < y$ or $y < x$ holds (this is also know as the *trichotomy* property of linear orders).

1.2.0.33 Example. A standard example of a total order is $<$ on $\mathbb{N}$. A standard example of a nontotal (nonlinear) order is $\subset$ on 2^A. For example, taking as $A = \{0, 1\}$, we see that $\{0\}$ and $\{1\}$ are not comparable under $\subset$. That the latter is an order is trivial to verify (it is irreflexive by definition), a task that we leave to the reader. □

1.3 BIG AND SMALL INFINITE SETS; DIAGONALIZATION

Two broad distinctions of sets by size are *finite* vs. *infinite*. Intuitively, we can count the elements of a finite set and come up with a (natural) number at some distinct point in (future) time. No such possibility is open for infinite sets. Just as finite sets come in various sizes, a 5-element set, vs. a 0-element set, vs. a $2^{3500000}$-element set,

Cantor has taught us that infinite sets also come in various sizes. The *technique* he used to so demonstrate is of interest to us, as it applies to computability, and is the key topic of this section.

1.3.0.34 Definition. (Finite sets) A set A is *finite* iff it is either empty, or is in 1-1 correspondence with $\{x \in \mathbb{N} : x \leq n\}$. We prefer to refer to this "normalized" finite set by the sloppy notation $\{0, \ldots, n\}$.

In this case we say that "A has $n+1$ elements". If $A = \emptyset$ we say that "A has 0 elements". If a set is *not* finite, then it is *infinite*. □

1.3.0.35 Example. If A and B have $n+1$ elements, then $A \sim B$ (cf. Exercise 1.8.31). □

1.3.0.36 Theorem. *If* $X \subset \{0, \ldots, n\}$, *then there is no* onto *function* $f : X \to \{0, \ldots, n\}$.

Proof. First off, the claim holds if $X = \emptyset$, since then $f = \emptyset$ and its range is empty. Let us otherwise proceed by way of contradiction, and assume that it *is* possible to have such onto functions, for some n. Suppose then that the smallest n that allows this to happen is n_0, and let X_0 be a corresponding set "X" that works, that is, we have an onto $f : X_0 \to \{0, \ldots, n_0\}$. Thus $X_0 \neq \emptyset$, by the preceding remark, and therefore $n_0 > 0$, since otherwise $X_0 = \emptyset$.

Let us set $H = f_{\leftarrow}(\{n_0\})$. $\emptyset \neq H \subseteq X_0$; the $\neq$ by ontoness.

Case 1. $n_0 \in H$. Then $f \restriction (X_0 - H)$ is onto, from $X_0 - H$ to $\{0, \ldots, n_0 - 1\}$—where $X_0 - H \subset \{0, \ldots, n_0 - 1\}$—contradicting minimality of n_0.

Case 2. $n_0 \notin H$. If $n_0 \notin X_0$, then we are back to Case 1. Otherwise, $X_0 - H \not\subset \{0, \ldots, n_0 - 1\}$ and we need a bit more work to get a $Y \subset \{0, \ldots, n_0 - 1\}$, and an onto function from left to right, to get our contradiction.

Well, we first look at the subcase where $f(n_0) \uparrow$: then just ignore n_0; that is, take $Y = X_0 - H - \{n_0\}$. Our function (onto $\{0, \ldots, n_0 - 1\}$) is $f \restriction Y$.

Finally, consider the subcase where $f(n_0) = m$. Take $g = \Big(f - \big(\{\langle n_0, m\rangle\} \cup H \times \{n_0\}\big)\Big) \cup (H \times \{m\})$. Essentially, g is f; except that it ensures that (a) we get no output n_0, (b) $n_0 \notin \mathrm{dom}(g)$, and yet (c) we do obtain output m—to maintain ontoness. Now, taking $Y = X_0 - \{n_0\}$ we see that $g : Y \to \{0, \ldots, n_0 - 1\}$ is onto. □

1.3.0.37 Corollary. (Pigeon-Hole Principle) *If* $m < n$, *then* $\{0, \ldots, m\} \not\sim \{0, \ldots, n\}$.

Proof. If the conclusion fails then we have an onto $f : \{0, \ldots, m\} \to \{0, \ldots, n\}$, contradicting 1.3.0.36. □

Here is a "quick proof" of 1.3.0.37 that does not utilize 1.3.0.36: Since $A \sim A$ for any non-empty set, $\{0, \ldots, m\}$ has $m + 1$ elements. If $\{0, \ldots, m\} \sim \{0, \ldots, n\}$, then, by 1.3.0.34, it also has $n + 1$ elements. Impossible!

Pause. Do you accept this "proof"?◀

You shouldn't. "A has $n+1$ elements" is just informal jargon for "$A \sim \{0,\ldots,n\}$". It may well be that this naming was unfortunate, and that it *fails to* uniquely *assign a* number *to a finite set as "the number of its elements"*. That the nomenclature in quotes is apt is the content of Corollary 1.3.0.37, not the other way around.

1.3.0.38 Corollary. *There is no onto function from* $\{0,\ldots,n\}$ *to* $\mathbb{N}$.

"For all $n \in \mathbb{N}$" is, of course, implied (cf. 1.1.1.10).

Proof. Fix an n. By way of contradiction, let $g : \{0,\ldots,n\} \to \mathbb{N}$ be onto. The function f given below is onto from $\mathbb{N}$ to $\{0,\ldots,n+1\}$

$$\begin{array}{ll} f(0) & = 0 \\ f(1) & = 1 \\ f(2) & = 2 \\ & \vdots \\ f(n+1) & = n+1 \\ f(m) & = 0, \text{ for all } m > n+1 \end{array}$$

Thus (cf. Exercise 1.8.34) $(fg) : \{0,\ldots,n\} \to \{0,\ldots,n+1\}$ is onto, contradicting 1.3.0.36. □

Our mathematical definitions have led to what we hoped they would: That $\mathbb{N}$ is infinite!

$\mathbb{N}$ is a "canonical" infinite set, and sets that can be enumerated using natural number indices

$$a_0, a_1, \ldots$$

have a special name.

1.3.0.39 Definition. (Countable Sets) A set A is *countable*, if it is empty or (in the opposite case) if there is a way to arrange all its members in an infinite sequence, in a "row of locations", utilizing one location for each member of $\mathbb{N}$. It *is* allowed to repeatedly list any element of A, so that finite sets are countable. For example, $\{1\}$:

$$1, 1, 1, \ldots$$

Technically, this enumeration is a *total and onto* function $f : \mathbb{N} \to A$. We say that $f(n)$ is the nth element of A in the enumeration f. We often write f_n instead of $f(n)$ and then call n a "subscript" or "index". □

A closely related notion is that of a set that can be enumerated using the elements of $\mathbb{N}$ as indices, but *without repetitions*.

1.3.0.40 Definition. (Enumerable Sets) A set A is *enumerable* iff it is in 1-1 correspondence with $\mathbb{N}$. □

1.3.0.41 Example. Every enumerable set is countable, but the converse fails. For example, $\{1\}$ is countable but not enumerable due to 1.3.0.38. $\{2n : n \in \mathbb{N}\}$ is enumerable, with $f(n) = 2n$ effecting the 1-1 correspondence $f : \mathbb{N} \to \{2n : n \in \mathbb{N}\}$. □

1.3.0.42 Theorem. *If A is an infinite subset of $\mathbb{N}$, then $A \sim \mathbb{N}$.*

Proof. We will build a 1-1 and total enumeration of A, presented in a finite manner as a (pseudo) program below:

X $\leftarrow A$
n $\leftarrow 0$
repeat forever:
pick a, **the smallest member of** X
tag a **with** n **as a subscript; print** a_n
n $\leftarrow n + 1$
X $\leftarrow X - \{a_n\}$

Since A in not finite, this process never ends. In particular, all the members of A will be picked (picking always the smallest avoids gaps) and all numbers from $\mathbb{N}$ will be utilized as indices, considering the non-ending nature of the process, the sequential choice of indices, and the starting point $n = 0$. That is, the function $f : \mathbb{N} \to A$, given for all n by $f(n) = a_n$, is total and onto. Since f is strictly increasing—$f(n) < f(n+1)$—it is 1-1 (distinct inputs cause distinct outputs). □

See also Exercise 1.8.35.

1.3.0.43 Theorem. *Every infinite countable set is enumerable.*

Proof. Let $f : \mathbb{N} \to A$ be onto and total, where A is infinite. Let $g : A \to \mathbb{N}$ such that $(fg) = \mathbf{1}_A$ (1.2.0.27). Let us set $B = \text{ran}(g)$. Thus, g is onto B, and by 1.2.0.22 is also 1-1 and total. Thus it is a 1-1 correspondence $g : A \to B$, or $A \sim B$.

B *must* be infinite, otherwise (1.3.0.34), for some n, $A \sim B \sim \{0, \ldots, n\}$. By transitivity of $\sim$ (Exercise 31), this proves that A is finite, contradicting the hypothesis. Thus, by 1.3.0.42, $A \sim B \sim \mathbb{N}$, hence (again, Exercise 1.8.31) A is enumerable. □

So, if we can enumerate an infinite set at all, then we can enumerate it without repetitions. It is useful to observe that we can convert a multirow enumeration

$$(f_{i,j})_{\text{for all } i,j \text{ in } \mathbb{N}}$$

into a single-row enumeration quite easily. This is shown diagrammatically below. The "linearization" or "unfolding" of the infinite matrix of rows is effected by walking

along the arrows.

$$\begin{array}{ccccccc} (0,0) & & (0,1) & & (0,2) & & (0,3) \quad \ldots \\ & \nearrow & & \nearrow & & \nearrow & \\ (1,0) & & (1,1) & & (1,2) & & \\ & \nearrow & & \nearrow & & & \\ (2,0) & & (2,1) & & & & \\ & \nearrow & & & & & \\ (3,0) & & & & & & \\ \vdots & & & & & & \end{array}$$

Technically, the set $\mathbb{N} \times \mathbb{N}$—the set of "double subscripts" (i, j)—is countable. This can be seen by a less informal argument; in fact, $\mathbb{N} \times \mathbb{N} \sim \mathbb{N}$:

Perhaps the simplest way to see this is to consider the function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ given by $f(m, n) = 2^m 3^n$. It is clearly total, and (less) clearly 1-1: For the latter just show that

$$2^m 3^n = 2^{m'} 3^{n'} \text{ implies } m = m' \text{ and } n = n'$$

But $\text{ran}(f)$ is infinite (see Exercise 1.8.36). Thus $\mathbb{N} \times \mathbb{N} \sim \text{ran}(f) \sim \mathbb{N}$.

This unfolding of a matrix into a straight line yields a very useful fact regarding strings over countable sets (alphabets):

If the string alphabet V is countable, then the set of all strings *of length* 2 over V is also countable. Why? Because the arbitrary string of length 2 is of the form $d_i d_j$, where d_i and d_j represent the ith and j elements of the enumeration of V, respectively. Unfolding the infinite matrix exactly as above we get a single-row enumeration of these strings.

By induction on the length $n \geq 2$ of strings we see that the set of strings of any length $n \geq 2$ is also countable. Indeed, a string of length $n + 1$ is a string ab, where a has length n and $b \in V$. By the induction hypothesis, the set of all strings a can be arranged in a single row (is countable), and we are done exactly as in the case of the $d_i d_j$ above (think of d_i as an "a" and d_j as a "b").

Finally, let us collect *all* the strings over V into a set S. Is S countable? Yes! We can arrange S, at first, into an infinite matrix of strings $m_{i,j}$, that is, the *jth string of length i*. Then we employ our matrix-unfolding trick above.

Given what we understand as a "string" (cf. subsection 1.1.3), the above argument translates as

(1) If V is countable, then so is V^n for any $n \geq 2$.

(2) If V is countable, then so is V^+

With little additional effort one can see that if A and B are countable, then so is $A \times B$ and generalize to the case of $\bigtimes_{i=1}^{n} A_i$.

1.3.0.44 Remark. Let us collect a few more remarks on countable sets here. Suppose now that we start with a countable set A. Is every subset of A countable? Yes, because the composition of onto functions is onto (Exercise 1.8.34). As a special case, if A is countable, then so is $A \cap B$ for any B, since $A \cap B \subseteq A$.

In particular, there is only an enumerable set of formulae if we start with a countable alphabet V, since the set of formulae is a subset of V^+. This comment relates to the discussion under the **Pause** on p. 41.

How about $A \cup B$? If both A and B are countable, then so is $A \cup B$. Indeed, and without inventing a new technique, let

$$a_0, a_1, \ldots$$

be an enumeration of A and

$$b_0, b_1, \ldots$$

for B. Now form an infinite matrix with the A-enumeration as the 1st row, while every other row is the same as the B-enumeration. Now unfold this matrix!

Of course, we may alternatively adapt the unfolding technique to an infinite matrix of two rows. □

1.3.0.45 Example. Suppose we have a 3×3 matrix

$$\begin{matrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{matrix}$$

and we are asked: Find a sequence of three numbers, *using only* 0 *or* 1, that does not *fit* as a row of the above matrix—i.e., is *different from all rows*.

Sure, you reply: Take 0 0 0.

That is correct. But what if the matrix were big, say, $10^{350000} \times 10^{350000}$, or even *infinite*?

Is there a *finitely describable technique* that can produce an "unfit" row for any square matrix, even an infinite one? Yes, Cantor's *diagonal method* or technique.

He noticed that any row that fits in the matrix as the, say, i-th row, intersects the main diagonal at the same spot that the i-th column does.

Thus if we take the main diagonal—a sequence that has the same length as any row—and *change every one of its entries*, then it will not fit anywhere as a row! Because no row can have an entry that is different than the entry at the location where it intersects the main diagonal!

This idea would give the answer 0 1 0 to our original question. While 1000 11 3 also follows the principle and works, we were constrained by "using only 0 or 1". More seriously, in a case of a very large or infinite matrix it is best to have a simple technique that works even if we do not know much about the elements of the matrix. Read on! □

1.3.0.46 Example. We have an infinite matrix of 0-1 entries. Can we produce an infinite sequence of 0-1 entries that does not match any row in the matrix? Yes, take the main diagonal and flip every entry (0 to 1; 1 to 0).

If the diagonal has an a in row i, the constructed row has an $1 - a$ in column i, so it will not fit as row i. So it fits nowhere, i being arbitrary. □

1.3.0.47 Example. (Cantor) Let S denote the set of all infinite sequences of 0s and 1s.

Pause. What is an *infinite sequence*? Our intuitive understanding of the term is captured mathematically by the concept of a total function f with left field (and hence domain) $\mathbb{N}$. The n-th member of the sequence is $f(n)$.◀

Can we arrange *all* of S in an infinite matrix—one element per row? No, since the preceding example shows that we would miss at least one infinite sequence (i.e., we would fail to list it as a row), for a sequence of infinitely many 0s and/or 1s can be found, that does not match any row!

But arranging all members of S as an infinite matrix—one element per row—is tantamount to saying that we can enumerate all the members of S using members of $\mathbb{N}$ as indices.

So we cannot do that. S is not countable! □

1.3.0.48 Definition. (Uncountable Sets) A set that is not countable is called *uncountable.* □

So, an uncountable set is neither finite, nor enumerable. The first observation makes it infinite, the second makes it "more infinite" than the set of natural numbers since it is not in 1-1 correspondence with $\mathbb{N}$ (else it would be enumerable, hence countable) nor with a subset of $\mathbb{N}$: If the latter, our uncountable set would be finite or enumerable (which is absurd) according as it is in 1-1 correspondence with a finite subset or an infinite subset (cf. 1.3.0.42 and Exercise 1.8.31).

Example 1.3.0.47 shows that uncountable sets exist. Here is a more interesting one.

1.3.0.49 Example. (Cantor) The set of real numbers in the interval

$$(0,1] \stackrel{\text{Def}}{=} \{x \in \mathbb{R} : 0 < x \leq 1\}$$

is uncountable. This is done via an elaboration of the argument in 1.3.0.47.

Think of a member of $(0,1]$, *in form*, as an infinite sequence of numbers from the set $\{0,1,2,3,4,5,6,7,8,9\}$ prefixed with a dot; that is, think of the number's decimal notation.

Some numbers have representations that end in 0s after a certain point. We call these representations *finite*. Every such number has also an "infinite representation" since the non zero digit d immediately to the left of the infinite tail of 0s can be converted to $d - 1$, and the infinite tail into 9s, without changing the value of the number.

Disallow all finite representations.

Assume now by way of contradiction that a listing of all members of $(0, 1]$ exists, listing them via their infinite representations

$$\begin{array}{l} .a_{00}a_{01}a_{02}a_{03}a_{04}\cdots \\ .a_{10}a_{11}a_{12}a_{13}a_{14}\cdots \\ .a_{20}a_{21}a_{22}a_{23}a_{24}\cdots \\ .a_{30}a_{31}a_{32}a_{33}a_{34}\cdots \\ \vdots \end{array}$$

The argument from 1.3.0.47 can be easily modified to get a "row that does not fit", that is, a representation

$$.d_0d_1d_2\cdots$$

not in the listing.

Well, just let

$$d_i = \begin{cases} 2 & \text{if } a_{ii} = 0 \vee a_{ii} = 1 \\ 1 & \text{otherwise} \end{cases}$$

Clearly $.d_0d_1d_2\cdots$ does not fit in any row i as it differs from the expected digit at the i-th decimal place: should be a_{ii}, but $d_i \neq a_{ii}$. It is, on the other hand, an infinite decimal expansion, being devoid of zeros, and thus *should* be listed. This contradiction settles the issue. □

1.3.0.50 Example. (1.3.0.47 Revisited) Consider the set of all total functions from $\mathbb{N}$ to $\{0, 1\}$. Is this countable?

Well, if there is an enumeration of these one-variable functions

$$f_0, f_1, f_2, f_3, \ldots \tag{1}$$

consider the function $g : \mathbb{N} \to \{0, 1\}$ given by $g(x) = 1 - f_x(x)$. Clearly, this *must* appear in the listing (1) since it has the correct left and right fields, and is total.

Too bad! If $g = f_i$ then $g(i) = f_i(i)$. By definition, it is also $1 - f_i(i)$. A contradiction.

This is a "mathematized" version of 1.3.0.47; as already noted, an infinite sequence of 0s and 1s is just a total function from $\mathbb{N}$ to $\{0, 1\}$. □

The same argument as above shows that the set of all functions from $\mathbb{N}$ to itself is uncountable. Taking $g(x) = f_x(x) + 1$ also works here to "systematically change the diagonal" $f_0(0), f_1(1), \ldots$ since we are not constrained to keep the function values in $\{0, 1\}$.

1.3.0.51 Remark. Worth Emphasizing. Here is how we constructed g: We have a list of *in principle available indices* for g. We want to make sure that *none applies*. A convenient method to do that is to inspect each available index, i, and using the diagonal method do this: Ensure that g differs from f_i at input i, setting $g(i) = 1 - f_i(i)$.

This ensures that $g \neq f_i$; period. We say that *we cancelled the index i* as a possible "f-index" of g.

Since the process is applied *for each i, we have cancelled all possible indices for g*: For no i can we have $g = f_i$. □

1.3.0.52 Example. (Cantor) What about the set of all subsets of $\mathbb{N}$ —$\mathcal{P}(\mathbb{N})$ or $2^{\mathbb{N}}$?

Cantor showed that this is uncountable as well: If not, we have an enumeration of its members as

$$S_0, S_1, S_2, \ldots \tag{1}$$

Define the set

$$D \stackrel{\text{Def}}{=} \{x \in \mathbb{N} : x \notin S_x\} \tag{2}$$

So, $D \subseteq \mathbb{N}$, thus it must appear in the list (1) as an S_i. But then $i \in D$ iff $i \in S_i$ by virtue of $D = S_i$. However, also $i \in D$ iff $i \notin S_i$ by (2). This contradiction establishes that $2^{\mathbb{N}}$ is uncountable.

In particular, it establishes that D is not an S_i. □

1.3.0.53 Example. (Characteristic functions) First a definition: Given a set S in the context of a reference set $\mathbf{U}$, the characteristic function of S, denoted by χ_S, is given by

$$\chi_S(x) = \begin{cases} 0 & \textbf{if } x \in S \\ 1 & \textbf{if } x \in \overline{S} \end{cases}$$

If the reference set is $\mathbb{N}$, the characteristic function of $S \subseteq \mathbb{N}$ is

$$\chi_S(x) = \begin{cases} 0 & \textbf{if } x \in S \\ 1 & \textbf{if } x \in \mathbb{N} - S \end{cases}$$

Note that there is a 1-1 correspondence F between subsets of $\mathbb{N}$ and 0-1-valued functions from $\mathbb{N}$ simply given by $F(S) = \chi_S$—cf. Exercise 1.8.37. Thus

The set of 0-1-valued functions from $\mathbb{N}$ is in 1-1 correspondence with $\mathcal{P}(\mathbb{N})$

In particular, the concept of characteristic functions shows that Example 1.3.0.52 fits the diagonalization methodology. Indeed, $\chi_D(x) = 1 - \chi_{S_x}(x)$ for all x. In other words, χ_D is nothing else but the *altered* "main diagonal" (in bold face type) of the infinite matrix

$$\begin{array}{ccccc} \mathbf{S_0(0)} & S_0(1) & S_0(2) & S_0(3) & \ldots \\ S_1(0) & \mathbf{S_1(1)} & S_1(2) & S_1(3) & \ldots \\ S_2(0) & S_2(1) & \mathbf{S_2(2)} & S_2(3) & \ldots \\ \vdots & \vdots & \vdots & \vdots & \end{array}$$

□

1.3.0.54 Example. By Exercise 1.8.38 we have that $2^{\mathbb{N}\times\mathbb{N}} \sim 2^{\mathbb{N}}$ so that

The set of all subsets of $\mathbb{N} \times \mathbb{N}$ is uncountable

The above can be rephrased to

The set of all binary relations on $\mathbb{N}$ is uncountable

Thus, if we build our formulae with symbols out of a countable alphabet, then we *do not have enough symbols* to represent (in our notation) *all* binary relations on $\mathbb{N}$ by formulae. This observation concludes our discussion that started on p. 41, following Definition 1.2.0.4 and continued in 1.3.0.44. □

1.3.0.55 Example. (Russell's Paradox is a Diagonalization) Russell formed the collection of sets x given as

$$R = \{x : x \notin x\} \tag{1}$$

He argued that it is contradictory to accept R as a *set*: For if it is, and given that (1) is equivalent to the statement (for all sets x)

$$x \in R \equiv x \notin x \tag{2}$$

we can substitute the specific set R into the set variable x to obtain—from the truth of (2)—the truth of the special case

$$R \in R \equiv R \notin R$$

This, of course, is absurd!

Let us now argue *intuitively*—taking liberties with working with *all sets* at once!—that the above argument is *a diagonalization over all sets.*

Imagine an infinite matrix, M, whose columns and rows are labeled by *all* sets, arranged in the same order along rows and columns. Assume that the matrix has as entries only the numbers 0 and 1, entered such that in the location determined by the row (named) x and the column (named) y we have a 0 iff $y \in x$ is true (we have 1 otherwise). That is,

$$y \in x \text{ iff } M(x, y) = 0 \tag{1}$$

It follows that *each row represents a set as an array of 0s and 1s*—that is, as the set's *characteristic function.*[43]

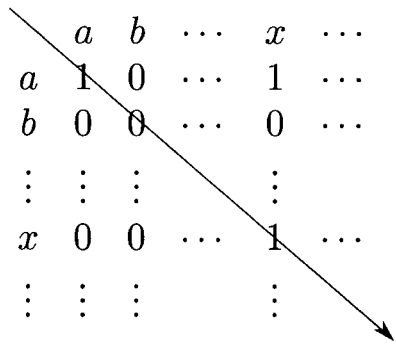
Thus, the partial depiction of the row for set a informs us that the following are true: $a \notin a$, $b \in a$ and $x \notin a$. Indeed, any array, X, of 0s and 1s whose entries are *labeled by the column names* represents a *collection* of sets that has y as a member iff the y-th entry of X is 0. For example, the diagonal collection

$$\begin{array}{c} \quad a\, b \quad x \\ \quad \downarrow\downarrow \quad \downarrow \\ d = 1\,0 \cdots 1 \cdots \end{array}$$

[43]Recall that this is an intuitive argument showing the (ironic) indebtedness of Russell's argument to Cantor's original diagonalization method. Thus we will not be splitting hairs about qualms such as: "Hmm, is this characteristic function defined over the collection of *all sets*? Can we do that?" Yes, because this is only a qualitative argument to tease out the diagonal argument that was hidden in Russell's proof.

contains b, but neither a, nor x.

The matrix M



Let us do Cantor's trick now: We take the main diagonal d and form an array $\widetilde{d}$ from it, by swapping *all* 1s with 0s. This $\widetilde{d}$ cannot fit as a row anywhere in the matrix M since it will disagree at the diagonal entry in any placement.

The fact that the collection of sets (named) $\widetilde{d}$ does *not* fit as a row of M means that *it is not a set*—because all sets are accounted for as row labels in M!

But which collection does $\widetilde{d}$ represent?

Well, using the analogy of X above, y is in $\widetilde{d}$ iff the y-th entry of $\widetilde{d}$ is 0 iff the y-th entry of d is 1 iff $y \notin y$. Thus,

$$\widetilde{d} = R, \text{ Russell's "paradoxical collection"}$$

□ 

1.4 INDUCTION FROM A USER'S PERSPECTIVE

In this section we will review the two widely used forms of induction, *complete* (or strong) induction (also called *course-of-values* induction) and *simple* induction. We will see how they are utilized, and when one is more convenient than the other; relate them to each other, but also to another principle that is valid on natural numbers, the *least (integer) principle*.

1.4.1 Complete, or Course-of-Values, Induction

Suppose that $\mathscr{P}(n)$ is a "property"—that is, a *formula* of one free variable, n—of the natural number n. To prove that $\mathscr{P}(n)$ holds *for all* $n \in \mathbb{N}$ it suffices to prove *for the arbitrary* n that $\mathscr{P}(n)$ holds.

What we mean by "arbitrary" is that we do not offer the proof of $\mathscr{P}(n)$ for some specific n such as $n = 42$; or n even; or any n that has precisely 105 digits, etc. If the proof indeed has not cheated by using some property of n beyond the generic

"$n \in \mathbb{N}$", then our proof is *equally valid for any* $n \in \mathbb{N}$; we have succeeded in effect to prove $\mathscr{P}(n)$, *for all* $n \in \mathbb{N}$ (cf. 1.1.1.10 and 1.1.1.15).

1.4.1.1 Example. Suppose $\mathscr{P}(n)$ stands for the statement

$$0 + 1 + 2 + 3 + \cdots + 2n = n(2n + 1) \tag{1}$$

One way to prove (1), for all n, is as follows: Fix, but *do not specify*, n—that lack of specification makes it *arbitrary*. Note the pairs below—separated by semicolons—each consisting of two numbers that are equidistant from the two ends of the sequence $0, 1, 2, 3, \ldots, 2n$

$$0, 2n; 1, 2n - 1; 2, 2n - 2; \ldots; n, 2n - n$$

The above sequence is (almost) a permutation of the sequence $0, 1, 2, 3, \ldots, 2n$, hence the sum of its terms is the same as the left hand side of (1), plus n.

Pause. Why "plus n"?◀

We have $n+1$ pairs, the sum of each being $2n$, thus the left hand side of (1) equals $(n + 1)2n - n$. An easy calculation shows that $(n + 1)2n - n = n(2n + 1)$. □

Now the above endeavor—proving some $\mathscr{P}(n)$ for the arbitrary n—is not always easy. *In fact, the above proof—attributed to Gauss—had a rabbit-off-a-hat flavor.* It would probably come as a surprise to the uninitiated that *we can pull an extra assumption out of the blue* and use it toward proving $\mathscr{P}(n)$, and not only that: When all is said and done, this process, *with* the extra assumption, is as good as if we have proved $\mathscr{P}(n)$ *without* the extra assumption!

This out-of-the-blue assumption is that

$$\mathscr{P}(k) \text{ holds for all } k < n \tag{I}$$

or, put another way, that *the history*, or the *course-of-values*, of $\mathscr{P}(n)$, namely,

$$\mathscr{P}(0), \mathscr{P}(1), \ldots, \mathscr{P}(n - 1) \tag{II}$$

holds—that is, it is a sequence of valid statements.

The extra assumption, (I) or (II), goes by the name *induction hypothesis* (I.H.) The technique of proving

$$\text{for all } n, \text{ we have that } \mathscr{P}(n) \text{ holds} \tag{2}$$

using an I.H. toward the proof, is called *proof by strong (complete, course-of-values) induction.*

The *application* (technique) of the proof by strong induction is:

(a) Pick an arbitrary n and prove the validity of $\mathscr{P}(n)$ having *also assumed the validity of (I) or (II).*

(b) Once step (a) is completed, we conclude (2).

We note that the history, (II), of $\mathscr{P}(n)$ is *empty* if $n = 0$. Thus every proof by strong induction has two cases to consider: the one where the history helps, *because it exists*, i.e., when we have $n > 0$; and the one where the history *does not help*, because it simply does not exist, i.e., when $n = 0$.

Thus, the application of strong induction morphs into a two-step method:

(A) Pick an arbitrary $n > 0$ and prove the validity of $\mathscr{P}(n)$, *having assumed the validity of (I) or (II).*

(B) For $n = 0$ prove $\mathscr{P}(n)$—i.e., $\mathscr{P}(0)$—directly.

(C) Once steps (A) and (B) are completed, we conclude (2).

Some jargon: As we noted, (I) or (II) are called the I.H. Step (A) above is called the *Induction Step* (I.S.). Step (B) is called the *basis* of the induction. The process (A)–(C) is *proof by induction on* n.

One often sees the basis done first, but it should be clear that it is just one of two cases to be considered, and the cases can be taken care of in any order.

It cannot be emphasized enough that the phrase "Pick an arbitrary $n > 0$ and prove..." is synonymous with "*Fix*, but do not specify, an $n > 0$ and prove ..."

Clearly the I.H. is for a *fixed but unspecified* n*—not* for all n, as the latter would beg the very question we are called to settle by induction!

1.4.1.2 Example. (Example 1.4.1.1 Revisited) We will prove (1) above, for all n, by strong induction, faithfully following the plan (A)–(B) above. Fix an arbitrary n. We have two cases:

Case $n > 0$. We assume the I.H. and try to prove (1). Well, we calculate as follows:

$$\begin{aligned} 0+1+2+\cdots+2(n-1)+2n-1+2n &= (n-1)\big(2(n-1)+1\big)+4n-1 \\ &= (n-1)(2n-1)+4n-1 \\ &= 2n^2-n-2n+1+4n-1 \\ &= 2n^2+n \\ &= n(2n+1) \end{aligned}$$

Note that the I.H. says

$$0+1+2+3+\cdots+2k = k(2k+1), \text{ for } k < n \tag{3}$$

This, in particular, is true for $k = n-1$, a fact we have used in the first calculation step above.

Case $n = 0$. In this case the statement to prove, namely, (1), becomes $0 = 0$, which is true. □

Hm. The I.H. (3) above seems to be an overkill given that only the case $k = n - 1$ was utilized in the I.S. Good point! We take it up in the next example.

1.4.1.3 Example. This time we prove, for all n,

$$\text{if } n > 1, \text{ then } n \text{ has a prime factor} \tag{4}$$

The reader will recall that a *factor* of n is a natural number m such that for some natural number k we have $n = mk$. A natural number p is a *prime* number (or just a prime) if and only if it is greater than 1, and all its factors are p and 1.

By strong induction, we take up first the case for an arbitrary (but fixed) n that allows a non-empty history; thus we assume the I.H. corresponding to (4):

$$\text{for all } k < n, \text{ if } k > 1, \text{ then } k \text{ has a prime factor} \qquad (I.H.)$$

The non-empty history case corresponds to $n \geq 3$, since $1 < k$ *and* $k < 2$ are inconsistent.

Let then $n \geq 3$. If n is a prime, then we are done (n is a factor of n). Alternatively, suppose that it is not. Then *there exist* a and b such that $n = ab$, where $a \neq 1 \neq b$—else n would be prime! Can $a < 1$? No, for then $a = 0$ and hence $n = 0$, contrary to the case we are in. Thus $a > 1$. Similarly $b > 1$. The latter yields $n = ab > a$.

Therefore the I.H. applies to a, that is, a has a prime factor, p. This means that for some m, $a = pm$. But then, $n = pmb$, and hence n has a prime factor.

The "basis" encompasses all the cases that have empty history: $n = 0, 1, 2$. For the first two the claim is vacuously satisfied as $n > 1$ is false. For $n = 2$ it is satisfied by virtue of 2 being a prime. □

This example shows the value of an I.H. that refers to the entire history below n: We have no way of controlling where a falls in the sequence $0, 1, 2, \ldots, n - 1$. It is unreasonable to expect that $a = n - 1$ in general. For example, if $n = 6$, then $a = 2$ and $b = 3$, or $a = 3$ and $b = 2$. But $n - 1 = 5$.

1.4.2 Simple Induction

Since on occasion we will also employ *simple* induction in this book, let me remind the reader that in this kind of induction the I.H. is not the assumption of validity of the entire history, but that of just $\mathscr{P}(n-1)$. As before, simple induction is carried out for the arbitrary n, so we need to work out two cases: when the I.H. exists ($n > 0$) and when it does not ($n = 0$). The case of proving $\mathscr{P}(0)$ directly is still called the *basis* of the (simple) induction.

The reader will notice that Example 1.4.1.2 can be recast under a simple induction proof since in the first step of the $n > 0$ case we only have used the assumption that (1) is true for $k = n - 1$.

Common practice has it that in performing simple induction the majority of users in the literature take as I.H. $\mathscr{P}(n)$ while the I.S. involves proving $\mathscr{P}(n+1)$.

1.4.3 The Least Principle

The least principle states that each non-empty subset of natural numbers contains a smallest (least) number.

1.4.3.1 Example. (Euclid) We prove that given a natural number $b > 1$, each natural number n can be expressed as $n = bq + r$ for some natural numbers q and r , where $0 \leq r < b$. *We only argue the case* $n > 0$ *since the case* $n = 0$ *is trivial:* $n = 0b + 0$.

So let $n > 0$. Note that the set $S = \{bx - n : x \in \mathbb{N} \wedge bx - n > 0\}$ is not empty. For example, since $bn > n$ (by $b > 1$), it is $bn - n > 0$. By the least principle, S contains a smallest number, which has the form $bm - n$ for some m. From $m \neq 0$ (since $-n$ is not positive and cannot be in S) we get $q = m - 1 \in \mathbb{N}$. Since $bq - n < bm - n$, it is $bq - n \notin S$. Thus $bq - n \leq 0$, i.e., $n - bq \geq 0$.

We set $r = n - bq$. Now, since $bq \leq n < b(q + 1)$ (recall, $m = q + 1$) we have $0 \leq n - bq < b(q + 1) - bq$, that is, $0 \leq r < b$. □

A related result that does not need the least principle (nor induction) is that the *quotient* q and *remainder* r are uniquely determined by n and b: Indeed, suppose that we have

$$n = bq' + r' \tag{5}$$

$$0 \leq r' < b \tag{6}$$

$$n = bq'' + r'' \tag{7}$$

$$0 \leq r'' < b \tag{8}$$

By (5) and (7) we have

$$b|q' - q''| = |r' - r''| \tag{9}$$

Can it be that $|q' - q''| \neq 0$? If so, $|q' - q''| \geq 1$, hence, multiplying both sides by b and using (9),

$$|r' - r''| \geq b \tag{10}$$

(6) and (8) tell a different story though! They yield [e.g., think of (8) as $-b < -r'' \leq 0$ and add with (6), term by term] $-b < r' - r'' < b$, that is $|r' - r''| < b$, which contradicts (10).

We thus must answer the earlier question "Can it be that $|q' - q''| \neq 0$?" by "no". But then (9) yields also $r' = r''$, as needed.

1.4.4 The Equivalence of Induction and the Least Principle

Somewhat surprisingly, all three proof techniques, by least principle, by simple or by course-of-values induction, have exactly the same power.

1.4.4.1 Theorem. *The least principle is equivalent to course-of-values induction.*

Proof. This proof requires two directions.

One, we can prove the least principle, *using strong induction*: Indeed, let $\emptyset \neq S \subseteq \mathbb{N}$. We will argue, by way of contradiction, that S has a least element.

So let instead S have *no* such element. The plan is to use strong induction to arrive at a contradiction. We may encounter more than one such contradictions, but the "primary" one that we will strive for is to prove that $S = \emptyset$—contrary to hypothesis—which is tantamount to $\mathbb{N} = \mathbb{N} - S$, or in many words:

$$\text{for all natural numbers } n,\ n \in \mathbb{N} - S \tag{1}$$

For the basis, we argue that $0 \in \mathbb{N} - S$. Indeed, if not, then $0 \in S$ will be least in S, contradicting what we assumed for S. Let then pick an $n > 0$ and accept as I.H. that for all $k < n$ we have $k \in \mathbb{N} - S$. It immediately follows that $n \in \mathbb{N} - S$ for otherwise it is the first n to enter S, which makes it least in S! We have proved (1).

Two, we prove that strong induction is valid, by *assuming that the least principle is*. That is, we will show the following, for any property $\mathscr{P}(n)$:

If $\mathscr{P}(0)$ holds, and if, for any $n > 0$, $\mathscr{P}(n)$ holds whenever all of $\mathscr{P}(0), \ldots, \mathscr{P}(n-1)$ hold; then $\mathscr{P}(n)$ holds for all n.

So we assume that the if-part of the italicized statement above is valid and prove the then-part, that "$\mathscr{P}(n)$ holds for all n".

Well, *assume we are wrong* in our conjectured conclusion (then-part). But then

$$S = \{n : \neg\mathscr{P}(n)\} \text{ is not empty.}$$

By the least principle, we have a smallest member of S, let us call it m. Now, $m \neq 0$, since the italicized statement's if-part includes the validity of $\mathscr{P}(0)$. What about $0, 1, 2, \ldots, m-1$ then? (Now that we know that $m - 1 \geq 0$, we may ask.) Well, none are in S (all being smaller than m), that is, they all satisfy $\mathscr{P}$. But then, the if-part of the italicized statement guarantees that $\mathscr{P}(m)$ must hold as well. This is no good because it says $m \notin S$!

This contradiction forces us to *backtrack over* our "*assume we are wrong*" above. So it is, after all, the case that $\mathscr{P}(n)$ *does* hold for all n. □

1.4.4.2 Theorem. *Simple induction and course-of-values induction have the same power.*

Proof. That is, one tool can simulate the other. We need to prove two things:

One, whatever property $\mathscr{P}(n)$ we can prove (for all n) via simple induction, we can also prove it *using strong induction*. Simple induction achieves this:

If $\mathscr{P}(0)$ holds, and if, for any $n > 0$, $\mathscr{P}(n)$ holds whenever $\mathscr{P}(n-1)$ holds; then $\mathscr{P}(n)$ holds for all n.

So *assume the if-part of the italicized statement*. Can *course-of-values induction* prove the then-part, namely, that "$\mathscr{P}(n)$ holds for all n"?

Well, strong induction will have to check that $\mathscr{P}(0)$ holds: That much is given by the if-part above. Now, for the arbitrary $n > 0$, strong induction's I.H. is that $\mathscr{P}(0), \ldots, \mathscr{P}(n-1)$ all hold. Can this assumption produce the truth of $\mathscr{P}(n)$? *Yes*,

because this strong I.H. yields the truth of $\mathscr{P}(n-1)$. By the if-part of the italicized statement above, this alone yields the truth of $\mathscr{P}(n)$.

Now, by strong induction, we indeed get the *then-part*: $\mathscr{P}(n)$ holds *for all* n.

Two, conversely, we prove that strong induction is valid, by *assuming that simple induction is*. That is, we will show that the following statement is valid, for any property $\mathscr{P}(n)$:

$$\begin{array}{l}\text{If } \mathscr{P}(n) \text{ holds on the assumption that, for all } k < n, \mathscr{P}(k) \text{ holds;} \\ \text{then } \mathscr{P}(n) \text{ holds for all } n.\end{array} \tag{2}$$

So we will *assume* the validity of if-part of (2), and then *employ simple induction* to *prove* the then-part, that

$$\mathscr{P}(n) \text{ holds for all } n \tag{3}$$

We will be a bit trickier here, so let us consider the new property $\mathscr{Q}(m)$ defined as follows:

$$\text{for all } k < m, \mathscr{P}(k) \text{ holds} \tag{4}$$

So, instead of directly proving (3),

$$\text{I will prove that, for all } n \in \mathbb{N}, \mathscr{Q}(n) \text{ holds} \tag{5}$$

I deliver on the promise (5) by simple induction, which, by assumption, is *the* tool at my disposal in this part of the proof: First, by (4), $\mathscr{Q}(0)$ says "for all $k < 0$, $\mathscr{P}(k)$ holds". I need to verify this, my (simple) induction's basis. Fortunately, the statement in quotes is *vacuously true* since it is impossible to refute it since a refutation requires a $k < 0$ [that makes $\mathscr{P}(k)$ false].

Next, let us fix an n and take the I.H. that $\mathscr{Q}(n)$ is true. We proceed to show that $\mathscr{Q}(n+1)$ is true too, and this will conclude (5). Now, $\mathscr{Q}(n+1)$ says "for all $k < n+1$, $\mathscr{P}(k)$ holds", or, "for all $k \leq n$, $\mathscr{P}(k)$ holds". Another way of putting it is: for all $k < n$, *and* for $k = n$, $\mathscr{P}(k)$ holds. That is, we want to show that

$$\mathscr{Q}(n) \text{ and } \mathscr{P}(n) \text{ hold} \tag{6}$$

Now $\mathscr{Q}(n)$ is true by the I.H. of our simple induction. That is, for all $k < n$, $\mathscr{P}(k)$ is true, by the definition of $\mathscr{Q}$ in (4). But we have assumed the if-part of (2), and this yields the truth of $\mathscr{P}(n)$. Thus (6) is established, i.e., $\mathscr{Q}(n+1)$, is true. Hence we have concluded (5). Having moreover just seen that $\mathscr{Q}(n)$ implies $\mathscr{P}(n)$, for any n, (5) implies that $\mathscr{P}(n)$ too holds for all n—and this statement is (3). □

At this point we can "strengthen" our inductions to "start" (basis) at any integer $n_0 > 0$.

Simple induction with non zero basis: To prove that, for all $n \geq n_0$, $\mathscr{P}(n)$ holds just do:

(A) Prove the truth of $\mathscr{P}(n_0)$.

(B) Fix an arbitrary $n \geq n_0$ and prove the truth of $\mathscr{P}(n+1)$ on the assumption that $\mathscr{P}(n)$ holds.

Strong induction with non zero basis: To prove that, for all $n \geq n_0$, $\mathscr{P}(n)$ holds just do:

(a) Prove the truth of $\mathscr{P}(n_0)$.

(b) Fix an arbitrary $n > n_0$ and prove the truth of $\mathscr{P}(n)$ on the assumption that $\mathscr{P}(k)$ holds for all $n_0 \leq k < n$.

1.4.4.3 Exercise. Start by the trivial observation that the least principle holds on the set $\mathbb{N}_{n_0} = \{n_0, n_0 + 1, n_0 + 2, \ldots\}$, namely: *Every non-empty subset of* $\mathbb{N}_{n_0}$ *has a least element.* Now modify the proof of 1.4.4.1 (using $\mathbb{N}_{n_0}$ instead of $\mathbb{N}$, judiciously) to conclude that the proof schema (a)–(b) above is equivalent to the least principle on the set $\mathbb{N}_{n_0}$.

Conclude that the proof schema (a)–(b) is valid. □

1.4.4.4 Exercise. Imitate the proof of 1.4.4.2 to prove that the schemata (a)–(b) and (A)–(B) above are equivalent in power.

Conclude that the proof schema (A)–(B) is valid. □

1.5 WHY INDUCTION TICKS

Induction is neat, but is it a valid principle? Why should we believe such a thing? Unfortunately, the previous section does not shed much light other than the somewhat surprising equivalence of the two induction principles with the least principle.

It turns out that we cannot prove either of the three as valid from any substantially simpler and therefore more readily believable facts of arithmetic. We can build a *plausible case*, however. Given the equivalence of the three, let us use simple induction as the pivot of our plausibility argument.

Simple induction is, intuitively, a proof generator that, for each given property $\mathscr{P}(n)$, certifies the latter's validity for any n that we want: Recall that the combination of the I.H. and I.S. establish *for the arbitrary* n, that *if* $\mathscr{P}(n)$ *is valid, then so is* $\mathscr{P}(n+1)$. Thus given the starting point, that is, the validity of $\mathscr{P}(0)$, we can certify the validity of $\mathscr{P}(1)$. And then of $\mathscr{P}(2)$. If we repeat this process—of inferring the truth of $\mathscr{P}(n+1)$ from that of $\mathscr{P}(n)$—for $n = 0, 1, 2, 3, \ldots, k-1$, for any k that we desire, then we will obtain the validity of $\mathscr{P}(k)$ (in k steps).

Imagine the process running for ever. Then the truth of $\mathscr{P}(n)$, for $n = 0, 1, 2, \ldots$ is established!

This argument is quite plausible, but glosses over two things: A mathematical proof has *finite length* so it cannot be an *infinite process* running for for $n = 0, 1, 2, \ldots$ Moreover, we must be sure that "for all $n \in \mathbb{N}$" really means *the same thing as* "for $n = 0, 1, 2, 3, \ldots$", or that $\mathbb{N}$ is *the smallest set around* with the properties[44]

(a) it contains 0

[44]Some bigger sets that have the properties (a) and (b) include $\mathbb{Z}$, the set of all integers; $\mathbb{Q}$, the set of all rational numbers; $\mathbb{R}$, the set of all real numbers; and more.

(b) if it contains n, it also contains $n+1$.

By the way, by "smallest" we mean that any other set T with the properties will satisfy $\mathbb{N} \subseteq T$.

Hm. This sounds right! $\mathbb{N}$ *is* the smallest set there is that satisfies (a) and (b), is it not? And if we are content with that, then here is a "real" proof of the simple induction principle, one that has *finite length*!

Pick any property $\mathscr{P}(n)$ and assume that we have performed the steps of simple induction, that is, we have already proved that

(A) $\mathscr{P}(0)$ is true.

(B) On the I.H. that $\mathscr{P}(n)$ is true we have proved that $\mathscr{P}(n+1)$ is true too.

Now let us form the set $S = \{n : \mathscr{P}(n)\}$. By (A), we have that $0 \in S$—that is, S satisfies (a) above. By (B), if $n \in S$, then also $n+1 \in S$—again, S satisfies (b) above. Since $\mathbb{N}$ is the smallest that satisfies (a) and (b), we have $\mathbb{N} \subseteq S$. That is, for all $n \in \mathbb{N}$ we have $n \in S$. Expressing this in terms of $\mathscr{P}(n)$ we have

$$\text{for all } n \in \mathbb{N},\ \mathscr{P}(n) \text{ holds} \tag{1}$$

That is, performing successfully the steps of simple induction—(A) and (B)—on $\mathscr{P}(n)$ we have succeeded in obtaining (1) as induction promises. Induction works!

Not so fast. Let us pick any set R that satisfies (a) and (b) above. I will show by induction that

$$\text{for all } n \in \mathbb{N},\ n \in R \text{ holds} \tag{2}$$

Well, the basis $0 \in R$ is satisfied, since R obeys (a). Let us fix an n now and take the I.H. $n \in R$. But, because R obeys (b), we will also have $n+1 \in R$. By simple induction, we have proved (2). But that says $\mathbb{N} \subseteq R$. Since R was *arbitrary* we have used induction to prove that $\mathbb{N}$ is the smallest set satisfying (a) and (b).

Thus the validity of induction and the just stated property of $\mathbb{N}$ are equivalent principles and we are back to square one: We have *not* succeeded in providing a proof of the validity of induction that is based *on more primitive, non equivalent to induction*, principles.

However, it is expected that our discussion brought some degree of comfort to the reader about the plausibility (and naturalness) of the induction principle! Mathematicians have long ago stopped worrying about this, and have adopted the induction principle as one of the starting points, i.e., nonlogical axioms, of (Peano) arithmetic.

1.6 INDUCTIVELY DEFINED SETS

One frequently encounters *inductive*—or, as they are increasingly frequently called, *recursive*—definitions of sets. This starts like this: Suppose that we start with the alphabet $\{0, 1\}$ and want to build strings as follows: We want to include ϵ, the empty

string. We also want the *rule* or *operation* that asks us to include $0A1$ if we know that the string A is included. So, some strings we might include are $\epsilon, 01, 0011$ and 001. The first was included outright, while the second and third are justified by the rule, via the presence of ϵ and 01, respectively. The last one would be legitimate if we knew that 0 was included. But is it? That is not a fair question. It becomes fair if we consider the *smallest*—with respect to inclusion $\subseteq$—set of strings that we can build, by including ϵ and repeatedly applying the rule. Then it can be proved that neither 0 nor 001 can be included in this smallest set.

There are several examples in mathematics and theoretical computer science of "smallest" sets defined from some start-up objects via a set of operations or rules whose application on existing objects yields new ones to include. Another one is the set of terms, formulae and proofs of logic. Further down we will encounter more examples such as the set of partial recursive and primitive recursive functions. But why look that far: Perhaps the simplest such smallest set built from initial objects and the application of operations is $\mathbb{N}$, as we have noted already: the initial object is 0 and the operation is "$+1$", the *successor function*.

The purpose of this section is to offer some unifying definitions and discuss their connection to each other.

1.6.0.5 Definition. (Operations) An n-ary *operation* or *rule* is a (binary) relation R such that whenever aRb, then a is an n-tuple. We will write $R(a_1, \ldots, a_n, b)$ rather than $R(\langle a_1, \ldots, a_n\rangle, b)$ or $\langle a_1, \ldots, a_n\rangle Rb$. We will call the sequence of objects $a_1, \ldots, a_n$ inputs, and the object b an *output*, or a *result* of R applied to the listed inputs.

It is *not* required that the relation be single-valued in its outputs. □

1.6.0.6 Definition. (Derivations) Given a set of objects, $\mathcal{I}$—the *initial* objects—and a set of operations $\mathcal{O}$. An $(\mathcal{I}, \mathcal{O})$-derivation, or just *derivation* if the context makes clear which $\mathcal{I}$ and $\mathcal{O}$ we have in mind, is a finite sequence of objects, $a_1, \ldots, a_n$ such that *every* a_i is one of

(1) a member of $\mathcal{I}$

(2) a result of some k-ary operation, from the set $\mathcal{O}$, applied on k inputs among the a_j that appear *before* a_i in the sequence —i.e., $j < i$ for all such inputs a_j.

We call the number n the *length of the derivation*. □

Since the legitimacy of any a_i in a derivation never depends on a a_k with $k > i$, it is clear that if $a_1, \ldots, a_r, \ldots, a_n$ is a derivation, then so is $a_1, \ldots, a_r$.

Note also that nowhere does the definition ask that the a_i be distinct. Indeed, once an a_i is placed as the i-th element, for the first time, it can be placed again thereafter—as $a_j = a_i$, with $j > i$—any number of times we wish. The same reason of legitimacy that applied originally to a_i still applies to all the additional placements a_j.

1.6.0.7 Example. Let $\mathcal{I} = \{0\}$ and $\mathcal{O}$ contain just the relation (given in atomic formula form) $x + 1 = y$, with y being the output variable. Then the reader can readily verify that the sequences

$$0, 0, 1, 2, 2, 2, 3, 0, 0, 4$$

and

$$0, 1, 2, 3, 4$$

are derivations. □

1.6.0.8 Example. Let $\mathcal{I} = \{0\}$ and $\mathcal{O}$ contain just the relation (given in atomic formula form) $x + 1 = y$, this time x being the output variable. Then the reader can readily verify that the sequences

$$0, 0, -1, -2, -2, -2, -3, 0, 0, -4$$

and

$$0, -1, -2, -3, -4, -5, -6$$

are derivations. □

1.6.0.9 Definition. A set S is *built by steps* from a set of initial objects, $\mathcal{I}$, and a set of operations $\mathcal{O}$ as follows: $S = \{a : a \text{ appears in some } (\mathcal{I}, \mathcal{O})\text{-derivation}\}$. □

If S is a set built by steps, then we can prove properties of its members by induction on the lengths of their derivations.

1.6.0.10 Example. Given the alphabet $\{0, 1\}$. Let $\mathcal{I} = \{\epsilon\}$, while $\mathcal{O}$ contains just the operation on strings $0x1 = y$—x being the input and y the output variables. We will show that the set S built by steps from the given pair $(\mathcal{I}, \mathcal{O})$ is $\{0^n 1^n : n \geq 0\}$, where, for any string A, and $n > 0$, A^n means

$$\underbrace{AA \cdots A}_{n \text{ copies of } A}$$

while A^0 means ϵ.

We have two directions to establish set equality:

$\subseteq$. For any $a \in S$ we do induction on its derivation length to show $a = 0^m 1^m$ for some m. If the length is 1, then it can only contain ϵ (initial object). Thus $a = 0^0 1^0$. We take as I.H. the truth of the claim when a is in a derivation of length $< n$.

For the I.H., suppose that a has a derivation, $a_1, \ldots, a_n$.

If $a = a_i$ with $i < n$, then since $a_1, \ldots, a_i$ is a shorter derivation, we are done by the I.H. If $a = a_n$ we have two cases: One, a is initial. This has already been dealt with. Two, $a = 0a_i1$, for some $i < n$.

By the I.H. $a_i = 0^m 1^m$. Thus, a has the same form.

$\supseteq$. For any $n \geq 0$ we prove that 0^n1^n *must* appear in some derivation. This is done by (simple) induction on n. For $n = 0$ (basis) $0^n1^n = \epsilon$; in S. Fix an n and assume that $0^n1^n \in S$ (this is the I.H.)

For the I.S. note that $0^{n+1}1^{n+1} = 00^n1^n1$. The I.H. guarantees a derivation exists in which 0^n1^n occurs. Without loss of generality (see remark following 1.6.0.6) the derivation has the form $a_1, a_2, \ldots, 0^n1^n$. This can be extended to the derivation $a_1, a_2, \ldots, 0^n1^n, 00^n1^n1$, hence $00^n1^n1 \in S$. □

1.6.0.11 Definition. A set S is *closed under an n-ary operation* iff, for every n-tuple of inputs chosen from S, all the results that the operation produces are also in S. □

For example, $\mathbb{N}$ is closed under $x + y = z$ (z is output), $x \times y = z$ (z is output), but not under $x - y = z$ (z is output). For example, $0 - 1 = -1 \notin \mathbb{N}$.

1.6.0.12 Definition. (Closure) Given a set of initial objects, $\mathcal{I}$, and a set of operations, $\mathcal{O}$. A set S is called the *closure of $\mathcal{I}$ under $\mathcal{O}$*—in symbols $S = \mathrm{Cl}(\mathcal{I}, \mathcal{O})$—iff it is the *smallest* set that contains $\mathcal{I}$ as a subset and is closed under *all* of the operations of $\mathcal{O}$.

A set such as $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ is also called *recursively* or *inductively* defined from the initial objects $\mathcal{I}$ and rules $\mathcal{O}$. □

Note that "smallest" means $\subseteq$-smallest, that is, if a set T contains $\mathcal{I}$ and is closed under $\mathcal{O}$, then $S \subseteq T$. This attribute, smallest, directly leads to the technique of (structural) induction over $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$:

Structural Induction: Let $S = \mathrm{Cl}(\mathcal{I}, \mathcal{O})$ and $\mathscr{P}(x)$ be a property (formula). To show that all $a \in S$ have the property, do the following:

(1) Prove $\mathscr{P}(a)$ for all $a \in \mathcal{I}$.

(2) Prove that the *property propagates* with every $R \in \mathcal{O}$, that is, whenever the inputs of R have the property, then so does the output.

The part "the inputs of R have the property" above is the I.H. for R. *There will be one I.H. for each $R \in \mathcal{O}$.* The I.S. for the R in question is to prove that, based on the I.H., the output has the property—i.e., the property propagates from the input side to the output side of the "black box" R.

Why "structural"? Because the induction is with respect to how the set was built.

The process (1)–(2) is *(structural) induction over S*, or *induction with respect to S*.

1.6.0.13 Theorem. *Structural Induction works: That is, if (1) and (2) above are indeed proved, then, for all $a \in S$, $\mathscr{P}(a)$ holds.*

Proof. Let $\mathbb{P} = \{a : \mathscr{P}(a) \text{ holds}\}$. Now (1) translates into $\mathcal{I} \subseteq \mathbb{P}$, while, by (2), for any $R \in \mathcal{O}$, whenever all the inputs of R are in $\mathbb{P}$ [i.e., they all satisfy $\mathscr{P}(x)$], then so is the output, that is, $\mathbb{P}$ is closed under *all* the operations of $\mathcal{O}$. By the "smallest" property of S (1.6.0.12), we have $S \subseteq \mathbb{P}$, that is, for all $a \in S$, $\mathscr{P}(a)$ holds. □

It turns out that not all properties $\mathscr{P}(x)$ lead to *sets* $\{x : \mathscr{P}(x)\}$—some such collections are "too big" to be, *technically*, "sets" (cf. Section 1.3).

Our proof above was within Cantor's informal or naïve set theory that glosses over such small print. However, formal set theory, that is meant to save us from our naïveté, upholds the "principle" of structural induction, (1)–(2), albeit using a slightly more complicated proof. Cf. Tourlakis (2003b).

1.6.0.14 Theorem. *Given a set of initial objects, $\mathcal{I}$, and a set of operations, $\mathcal{O}$. The two sets: S*—built by steps *(1.6.0.9) from $\mathcal{I}$ and $\mathcal{O}$—and* $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ *are equal.*

Proof. For $\subseteq$ we do induction on derivation length of $a \in S$. If the length equals 1, then $a \in \mathcal{I}$. Since $\mathcal{I} \subseteq \mathrm{Cl}(\mathcal{I}, \mathcal{O})$ by 1.6.0.12, the basis is settled. Assume next (I.H.) that for all $k < n$, if a occurs in a derivation of length k, then a is in the closure.

I.S.: Let a occur in a derivation of length n. If it does not occur at the right end, then the I.H. kicks in and a is in the closure. So let a be the last object in the derivation. If it is initial, we have nothing to add to what we said for the basis. Suppose instead that a is the result of an operation from $\mathcal{O}$ that was applied on inputs $a_{j_1}, \ldots, a_{j_r}$ that appeared to the left of a in the derivation. By the I.H. all the a_{j_m} are in the closure. The later being closed under all operations from $\mathcal{O}$ we conclude that the result of the operation, a, is in the closure.

For $\supseteq$ we do induction over $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$: For the basis, if $a \in \mathcal{I}$ then $a \in S$ via a derivation of length 1. We now show that the property "$a \in S$" propagates with every $R \in \mathcal{O}$. To unclutter exposition, and without loss of generality, fix an R—and pretend without loss of generality that its arity is 3—and let its inputs a, b, c be all in S. Let $R(a, b, c, d)$. We want $d \in S$.

Well, by I.H. there are three derivations $\ldots, a, \ldots$; $\ldots, b, \ldots$; and $\ldots, c, \ldots$

If we concatenate them into one sequence

$$\ldots, a, \ldots, \ldots, b, \ldots, \ldots, c, \ldots$$

we have a derivation (why?). Due to the way d is obtained, so is

$$\ldots, a, \ldots, \ldots, b, \ldots, \ldots, c, \ldots, d$$

But then $d \in S$ by the way S is obtained. □

1.6.0.15 Remark. The above is a significant theorem: If we want to prove properties of $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ as a whole, the best idea is to do structural induction over the set. If on the other hand we want to prove that some a is a member of $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$, then the best idea is to provide a derivation for it.

Compare: If we want to prove a property of all theorems of a theory, then we do induction over the theory that is built using 1.1.1.34 (and 1.1.1.38). If on the other hand we want to verify that a formula is a theorem, then we produce a proof for it. Evidently, by1.6.0.14 we have that the iterative definition of "theorem" in 1.1.1.34 is equivalent with the inductive one: *The set of all theorems is* $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ *where $\mathcal{I}$ is*

the adopted set of axioms and $\mathcal{O}$ is the adopted rules of inference. Cf. also 1.6.0.17 below.

> Note that since b appears in a derivation iff it is either initial or a *result* of some $R \in \mathcal{O}$ applied on *prior members of the derivation*—and the latter is tantamount to saying "*members of* $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$" because of 1.6.0.14—we have the following very useful "membership test":
>
> $b \in \mathrm{Cl}(\mathcal{I}, \mathcal{O})$ *iff* $b \in \mathcal{I}$ *or* b *is the result of some rule applied to members of* $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$.

In words, the theorem says that the inductive approach—forming the closure—and the iterative approach, building one element at a time via a derivation, yield the same result. □

1.6.0.16 Example. Let $\mathcal{I} = \{3\}$ and $\mathcal{O}$ consist of just $x + y = z$ and $x - y = z$, where in both cases z is the output variable. We are thinking of $\mathbb{Z}$ as our reference set here. Let us see why we have

$$\mathrm{Cl}(\mathcal{I}, \mathcal{O}) = \{3k : k \in \mathbb{Z}\} \tag{1}$$

For the $\subseteq$ we do, of course, induction over $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$. Well, $\mathcal{I}$ contains just 3, and $3 = 3 \times 1$, hence is in the right hand side (rhs) of (1).

Let us see that *membership to the right* propagates with the two rules: So let a and b be in the rhs. Then $a = 3m$ and $b = 3r$ for some m, r in $\mathbb{Z}$. Trivially, each of $a + b$ and $a - b$ is a multiple of 3.

As for $\supseteq$, let a be in the rhs, that is, $a = 3k$ for some $k \in Z$.

Case 1. $k \geq 0$. Let us do induction on $k \geq 0$ to show that $3k$ in the left hand side (lhs). Well, if $k = 0$, then we are done by the derivation $3, 0$ (why is this a derivation?).

Take as I.H. the truth of the claim for (fixed) k and go to $k + 1$. Given that $3(k + 1) = 3k + 3$, we are done by the I.H. and since the lhs is closed under $x + y = z$ (of course, 3 is in lhs).

Case 2. $k < 0$. Well, $0 \in \mathrm{Cl}(\mathcal{I}, \mathcal{O})$: indeed, apply $x - y = z$ to input $3, 3$. But then $3k \in \mathrm{Cl}(\mathcal{I}, \mathcal{O})$ as well, since $3k = 0 - 3(-k)$; now apply the same rule on inputs 0 and $3(-k)$ with the help of Case 1. □

1.6.0.17 Example. Let us work within arithmetic (simply for the sake of having a fixed alphabet of symbols). We take as $\mathcal{I}$ the set of all logical axioms (1.1.1.38), and these two rules form $\mathcal{O}$:

$$M(\mathscr{X}, \mathscr{Y}, \mathscr{Z}) \text{ holds iff } \mathscr{Y} \text{ has the form } \mathscr{X} \to \mathscr{Z} \qquad (MP)$$

and

$$G(\mathscr{X}, \mathscr{Y}) \text{ holds iff } \mathscr{Y} \text{ has the form } (\forall x)\mathscr{X} \text{ for some } x \qquad (Gen)$$

That is, our familiar MP and Gen. So, what is $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$? But of course—immediately from 1.6.0.14—it is the set of all absolute theorems (provable without nonlogical axioms) that we can prove if we employ as our *only* rules Gen and MP (cf. 1.1.1.34).

By induction over this $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$—or as logicians prefer to say, by *induction on theorems*—we can prove the soundness of this proof system: That every theorem, i.e., member of $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$, is true.

Well, the claim holds for $\mathcal{I}$ as we already know (1.1.1.39).

We only need to show that the claim propagates with the two rules above: Indeed, the MP is a special case of tautological implication, and Gen preserves truth by 1.1.1.15. □

In 1.1.1.34 we adopted all tautological implications—not just MP—as rules. This was expedient. It suffices to include just one such implication: MP. The interested reader can see why in Tourlakis (2008, 2003a).

Both examples 1.6.0.7 and 1.6.0.16 employ rules that are functions (single valued). Example 1.6.0.17 on the other hand has a rule that is not a function:

$$\text{input: } \mathscr{A}\text{; output: } (\forall x)\mathscr{A}$$

since for each of the infinitely many choices of x we have a different output (why "infinitely many"?)

A more crucial—and troublesome—observation is this: In 1.6.0.7 every member of the closure has a unique *immediate predecessor*. Not so in Examples 1.6.0.16 and 1.6.0.17. In the former, 12 could be $15 - 3$ or $6 + 6$ or $9 + 3$. Indeed, 3 is both initial, and something that can be (re)built: $6 - 3$, for example. In the latter example, if $\mathscr{A}, \mathscr{A} \to \mathscr{B}$ yield $\mathscr{B}$ so do infinitely many pairs $\mathscr{X}, \mathscr{X} \to \mathscr{B}$ for all possible choices of $\mathscr{X}$. This phenomenon is called *ambiguity*.

1.6.0.18 Definition. (Ambiguity) A pair $\mathcal{I}, \mathcal{O}$ is *ambiguous* if one or more of the following hold. Otherwise it is *unambiguous*.

(1) For some $a \in \mathrm{Cl}(\mathcal{I}, \mathcal{O})$ and some n-ary rule $R \in \mathcal{O}$, there are $\langle p_1, \ldots, p_n\rangle \neq \langle q_1, \ldots, q_n\rangle$ such that $R(p_1, \ldots, p_n, a)$ and $R(q_1, \ldots, q_n, a)$;

(2) For some $a \in \mathrm{Cl}(\mathcal{I}, \mathcal{O})$ and two distinct n-ary and m-ary rules R and Q in $\mathcal{O}$, there are $\langle p_1, \ldots, p_n\rangle$ and $\langle q_1, \ldots, q_m\rangle$ such that $R(p_1, \ldots, p_n, a)$ and $Q(q_1, \ldots, q_m, a)$;

(3) For some element $a \in \mathcal{I}$, there is an n-ary rule $R \in \mathcal{O}$, and a tuple $\langle p_1, \ldots, p_n\rangle$ such that $R(p_1, \ldots, p_n, a)$.

If $a \in \mathrm{Cl}(\mathcal{I}, \mathcal{O})$ and $R(p_1, \ldots, p_n, a)$ holds for some $R \in \mathcal{O}$, we will call $\langle p_1, \ldots, p_n\rangle$ a vector (or sequence) of *immediate predecessors* of a. For short, *i.p.* □

1.6.0.19 Example. Here is why ambiguity is trouble. Let us start with the alphabet of symbols $A = \{1, 2, 3, +, \times\}$. We will inductively define a restricted set of arithmetic expressions (for example, we employ no variables) as follows. Let $\mathcal{I} = \{1, 2, 3\}$ and let $\mathcal{O}$ consist of just two *string* operations:

$$\text{from strings } X \text{ and } Y \text{ form } X + Y \qquad (1)$$

$$\text{from strings } X \text{ and } Y \text{ form } X \times Y \tag{2}$$

Some examples are $1 + 1$, 2×1 and, more interestingly, $1 + 2 \times 3$. What do these strings *mean*? Let us assign the "natural" meaning: "1" means 1, "2" means 2, and "3" means 3. "+" means add (plus) and "×" means multiply. Thus, extending this to an arbitrary member of $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ we will opt for the natural approach: As $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ is defined inductively itself, why not effect a *recursive definition of meaning* via a function "EV" (for "evaluate"), which will compute the value of a member A of $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ *by calling itself recursively on A's i.p.*

Therefore, we define (if you will, we *program*) EV by:

$$\begin{aligned}
EV(1) &= 1\\
EV(2) &= 2\\
EV(3) &= 3\\
EV(X + Y) &= EV(X) + EV(Y)\\
EV(X \times Y) &= EV(X) \times EV(Y)
\end{aligned}$$

So what is the value (meaning) of $1 + 2 \times 3$? Well,

$$\begin{aligned}
EV(1 + 2 \times 3) &= EV(1 + 2) \times EV(3)\\
&= \Big(EV(1) + EV(2)\Big) \times 3\\
&= \Big(1 + 2\Big) \times 3\\
&= 9
\end{aligned}$$

No, no, you say. It is

$$\begin{aligned}
EV(1 + 2 \times 3) &= EV(1) + EV(2 \times 3)\\
&= EV(1) + \Big(EV(2) \times EV(3)\Big)\\
&= 1 + \Big(2 \times 3\Big)\\
&= 7
\end{aligned}$$

We are both "right", of course. The pair $\mathcal{I}, \mathcal{O}$ is ambiguous; in particular, the string $1+2\times 3$ has two i.p.: $\langle 1+2, 3\rangle$ on which the first computation is based, and $\langle 1, 2\times 3\rangle$ on which we based the second computation. □

While we are on the subject of closures, let us look at the very important *transitive closure* of a relation.

1.6.0.20 Definition. (Transitive Closure) The *transitive closure* of a relation R is the *smallest* (in the sense of inclusion, $\subseteq$) transitive relation that includes R, that is, if Q is *a* transitive closure of R, then we must have

(1) $R \subseteq Q$

(2) Q is transitive, and

(3) If T is transitive and $R \subseteq T$, then $Q \subseteq T$. □

(1) While we have no *a priori* reason to expect that transitive closures exist just by virtue of us coining this term, we can say one thing:

A relation R *cannot* have *more than one* transitive closure. Indeed, if Q and Q' are both transitive closures of R, then having Q' pose as "T" we get $Q \subseteq Q'$. Next, having Q so pose, we have $Q' \subseteq Q$. Thus, $Q = Q'$.

(2) Intuitively, we can imagine the (we can now say "*the*") transitive closure of a relation R as the relation that we get from R by step-by-step adding pairs $\langle a, c\rangle$ to the relation that we have built so far, as long as, for some b, $\langle a, b\rangle$ and $\langle b, c\rangle$ are already included. We stop this process of adding pairs as soon as we obtain a transitive relation via this process. This observation is made precise below.

1.6.0.21 Theorem. *For any relation R, its unique transitive closure* exists *and equals* $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$*, where $\mathcal{I} = R$—a set of ordered pairs—and $\mathcal{O}$ contains just one* operation on pairs *that, for any two input pairs $\langle a, b\rangle$ and $\langle b, c\rangle$ (note the common b), the operation produces the pair $\langle a, c\rangle$.*

We will denote the transitive closure of R by R^+.

Proof. We show that $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ satisfies (1)–(2) of 1.6.0.20, which will confirm that $R^+ = \mathrm{Cl}(\mathcal{I}, \mathcal{O})$. For (1), we are done by the property $\mathcal{I} \subseteq \mathrm{Cl}(\mathcal{I}, \mathcal{O})$ of any closure. For (2) we are done since $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ is closed under the operation in $\mathcal{O}$: If $\langle a, b\rangle$ and $\langle b, c\rangle$ are in $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$, then so is $\langle a, c\rangle$.

For (3), let T be transitive and $R \subseteq T$. We want to show that $\mathrm{Cl}(\mathcal{I}, \mathcal{O}) \subseteq T$. Well, both T and $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ are supersets of R and are closed under the operation "if $\langle a, b\rangle$ and $\langle b, c\rangle$ are included, then so is $\langle a, c\rangle$". But, as a closure, $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ is $\subseteq$-smallest with these two properties, therefore $\mathrm{Cl}(\mathcal{I}, \mathcal{O}) \subseteq T$ as needed. □

1.6.0.22 Corollary. *For any relation R, its transitive closure R^+ is equal to $\bigcup_{n\geq 1} R^n$.*

We also may write

$$R^+ = \bigcup_{n=1}^{\infty} R^n$$

Proof. Let us set $Q = \bigcup_{n=1}^{\infty} R^n$ and prove that $Q = \mathrm{Cl}(\mathcal{I}, \mathcal{O})$, where $\mathcal{I}, \mathcal{O}$ are as in 1.6.0.21.

For $Q \subseteq \mathrm{Cl}(\mathcal{I}, \mathcal{O})$ it suffices to prove that $R^n \subseteq \mathrm{Cl}(\mathcal{I}, \mathcal{O})$, for $n \geq 1$, by induction on n: For $n = 1$, $a\, R^1\, b$ means $a\, R\, b$ thus $\langle a, b\rangle \in \mathrm{Cl}(\mathcal{I}, \mathcal{O})$ since $R = \mathcal{I}$. Taking the obvious I.H. for n we next let $a\, R^{n+1}\, b$. This means that for some c we have $a\, R\, c$ and $c\, R^n\, b$. By the basis and I.H. respectively we have $\langle a, c\rangle$ and $\langle c, b\rangle$ in $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$, hence $\langle a, b\rangle$ is in $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ (transitivity).

For $Q \supseteq \mathrm{Cl}(\mathcal{I}, \mathcal{O})$ we do induction on the closure. Since $\mathcal{I} = R \subseteq Q$, we only need show that Q is transitive. Let then $a\, Q\, c$ and $c\, Q\, b$, hence, for some m and n, $a\, R^m\, c$ and $c\, R^n\, b$. Therefore $a\, R^m \circ R^n\, b$ and thus $a\, R^{m+n}\, b$ by Exercise 1.8.42. Thus $a\, Q\, b$. □

1.6.0.23 Remark. (1) Thus, we have $a\, R^+\, b$ iff, for some n, $a\, R^n\, b$ iff, for some sequence

$$a_0, \ldots, a_n$$

where $a_0 = a$ and $a_n = b$, we have

$$a_i R a_{i+1}, \text{ for } i = 0, 1, \ldots, n-1$$

The notation below is also common.

$$a\, R^+\, b \text{ iff, for some } a_j, \text{ it is } a\, R\, a_1\, R\, a_2\, R\, a_3 \cdots a_j\, R\, a_{j+1} \cdots a_{n-2}\, R\, a_{n-1} R\, b$$

(2) If R is on A, then its *reflexive transitive closure* is denoted by R^* and is defined by $\mathbf{1}_A \cup R^+$. That is, $a\, R^*\, b$ iff $a = b$ or $a\, R^+\, b$. □

1.7 RECURSIVE DEFINITIONS OF FUNCTIONS

We often encounter a definition of a function over the natural numbers such as

$$f(0, m) = 0$$
$$f(n+1, m) = f(n, m) + m$$

Is this a *legitimate* definition? That is, *is there* really a function *that satisfies* the above two equalities for all n and m? And if so, is there *only one* such function, or is the definition ambiguous? We address this question in this section through a somewhat more general related question.

1.7.0.24 Example. Let us look at a simpler question than the above and see if we can produce a good answer. First off, is there a function g given by the following two equalities for all values of n?

$$g(0) = 1$$
$$g(n+1) = 2^{g(n)}$$

Well, let's see: By induction on n we can show that $g(n) \downarrow$ for all n: Indeed, this is true for $n = 0$ by the first equality. Taking the I.H. that $g(n) \downarrow$ (for a fixed unspecified n) we can compute $g(n+1)$ so indeed

$$g(n) \downarrow \text{ for all } n$$

We can say then that the function g exists; right?

Wrong! A function is a set, in this case an infinite table of pairs (if we take its existence for granted). We did *not* show that it exists as *a table of pairs*; rather *we have only shown* that

If *a g satisfying the given equalities exists, then it is total.*[45]

[45] The set of texts on the subject of *the theory of computation*, which seriously propose the above erroneous "proof" of existence is non-empty.

We can however prove that any two functions satisfying the equalities must be equal. That is, if h is another such function, then $h = g$ or, on an input by input basis,

$$g(n) = h(n), \text{ for all } n \tag{1}$$

Note that we wrote $=$ in (1), rather than $\simeq$ (cf. 1.2.0.11), since we already know that if a function satisfies the given equalities, then it is total.

As for (1), for $n = 0$ we are done by the first equality. Taking as I.H. the case for some fixed n, the case for $n+1$ is easily settled: $g(n+1) = 2^{g(n)} = 2^{h(n)} = h(n+1)$, where the middle = is due to the I.H. □

So how do we settle existence? We *build* a function f (or g) given as above either iteratively, *by stages* [which is the usual approach in the literature, when done correctly; cf. Tourlakis (1984)] or as a *closure*; a set of the form $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ for appropriate $\mathcal{I}, \mathcal{O}$. The latter approach is from Tourlakis (2003b), which also develops the iterative approach. Mindful of the example in 1.6.0.19, we will define functions recursively on an inductively defined set $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ that is given via an unambiguous pair $\mathcal{I}, \mathcal{O}$.

1.7.0.25 Theorem. (Function definition by recursion) *Let $\mathcal{I}, \mathcal{O}$ be an* unambiguous *pair and $\mathrm{Cl}(\mathcal{I}, \mathcal{O}) \subseteq A$, for some set A. Let $h : \mathcal{I} \to B$ and $g_R : A \times B^r \to B$, for each r-ary $R \in \mathcal{O}$, be given functions.*[46]

Under these assumptions, **there is** *a* **unique** *function $f : \mathrm{Cl}(\mathcal{I}, \mathcal{O}) \to B$ such that*

$$y = f(x) \text{ iff } \begin{cases} y = h(x) & \text{and } x \in \mathcal{I} \\ \quad \text{OR,} \quad \text{for some } R \in \mathcal{O}, & \\ y = g_R(x, o_1, \ldots, o_r) & \text{and } R(a_1, \ldots, a_r, x) \text{ holds,} \\ \quad \text{where } o_i = f(a_i), \text{for } i = 1, \ldots, r & \end{cases} \tag{1}$$

The reader may wish to postpone studying this proof, but he should become thoroughly familiar with the statement of the theorem, and study the examples that follow the proof.

Proof. To *prove the existence* of a "solution" f to the given recursive definition (1), we will *build* a single-valued binary relation $F \subseteq A \times B$, which, when we rewrite $F(x, y)$ as "$y = F(x)$"—with y as the output variable—satisfies (1) above. To build it, we will realize it as an appropriate $\mathrm{Cl}(\mathcal{J}, \mathcal{T})$, for appropriately chosen initial set $\mathcal{J}$ and set of rules $\mathcal{T}$.

For each r-ary rule $R \in \mathcal{O}$, define the r-ary rule $\widehat{R}$ by

$$\widehat{R}\big(\langle a_1, o_1\rangle, \ldots, \langle a_r, o_r\rangle, \langle b, g_R(b, o_1, \ldots, o_r)\rangle\big) \text{ iff } R(a_1, \ldots, a_r, b) \tag{2}$$

[46] An r-ary operation (rule) is an $(r+1)$-ary relation; cf. 1.6.0.5.

For any $a_1, \ldots, a_r, b$, the above definition of $\widehat{R}$ is effected for all possible choices of $o_1, \ldots, o_r$ in B for which $g_R(b, o_1, \ldots, o_r)$ is defined.

By the way, there is no mystery in the definition of $\widehat{R}$ (the name chosen to show the close association with R): If we anticipate that (1) *does* have an f-solution, we can then view the o_i as the $f(a_i)$. Then $\widehat{R}$'s job is—*once we give it, in the form of input/output pairs, where the outputs are those of f on all the i.p. of b*—to compute $f(b)$ using g_R and to output the input/output pair $\langle b, f(b)\rangle$.

Collect now all the rules $\widehat{R}$ as defined, to form the rule-set $\mathcal{T}$.

As the initial objects-set $\mathcal{J}$, which we will associate with the rule-set $\mathcal{T}$, we take $\mathcal{J} = h$—that is, $\langle x, y\rangle \in \mathcal{J}$ iff $h(x) = y$.

Claim 1. The set $F = \mathrm{Cl}(\mathcal{J}, \mathcal{T})$ is a single-valued binary relation $\mathrm{Cl}(\mathcal{I}, \mathcal{O}) \to B$.

Proof of Claim 1. First off, that $F \subseteq \mathrm{Cl}(\mathcal{I}, \mathcal{O}) \times B$ is immediate: $\mathcal{J} \subseteq \mathrm{Cl}(\mathcal{I}, \mathcal{O}) \times B$, and each relation of $\mathcal{T}$ has as *output* a pair in $\mathrm{Cl}(\mathcal{I}, \mathcal{O}) \times B$, by definition (2).

We next establish that F is single valued in its second component, doing induction over $\mathrm{Cl}(\mathcal{J}, \mathcal{T})$. The claim to prove will be

$$\text{if } \langle a, b\rangle \in F \text{ and } \langle a, c\rangle \in F, \text{ then } b = c \qquad (*)$$

Basis: Suppose that $\langle a, b\rangle \in \mathcal{J}$ and let also $\langle a, c\rangle \in F$.

By 1.6.0.15, the latter entails, in *principle*:

(i) $\langle a, c\rangle \in \mathcal{J}$. Then $c = h(a) = b$,

OR,

(ii) for some r-ary $\widehat{R} \in \mathcal{T}$, we have $\widehat{R}(\langle a_1, o_1\rangle, \ldots, \langle a_r, o_r\rangle, \langle a, c\rangle)$, where $R(a_1, \ldots, a_r, a)$, $c = g_R(a, o_1, \ldots, o_r)$, and $\langle a_1, o_1\rangle, \ldots, \langle a_r, o_r\rangle$ are in F.

The right hand side of the capitalized "or" cannot be applicable—due to its requirement that $R(a_1, \ldots, a_r, a)$—given that $a \in \mathcal{I}$ and $(\mathcal{I}, \mathcal{O})$is unambiguous.

We next prove that the property $(*)$ propagates with each $\widehat{Q} \in \mathcal{T}$. So, let

$$\widehat{Q}(\langle a_1, o_1\rangle, \ldots, \langle a_r, o_r\rangle, \langle a, b\rangle)$$

Since by the previous argument $\langle a, c\rangle \notin \mathcal{J}$, let also

$$\widehat{P}(\langle a'_1, o'_1\rangle, \ldots, \langle a'_l, o'_l\rangle, \langle a, c\rangle)$$

where $Q(a_1, \ldots, a_r, a)$ and $P(a'_1, \ldots, a'_l, a)$, but also [cf. (2)]

$$b = g_Q(a, o_1, \ldots, o_r) \text{ and } c = g_P(a, o'_1, \ldots, o'_l) \qquad (3)$$

Since $(\mathcal{I}, \mathcal{O})$ is unambiguous, $Q = P$ (hence also $\widehat{Q} = \widehat{P}$); thus $r = l$, and $a_i = a'_i$, for $i = 1, \ldots, r$.

By I.H., $o_i = o'_i$, for $i = 1, \ldots, r$, hence $b = c$ by (3). *End of proof: Claim 1.*

Claim 2. F satisfies (1). Now that we know that F is a function we can write

$$b = F(a) \text{ for } \langle a, b\rangle \in F$$

Our task here is to show that if we replace the "(function) variable" f in (1) by the *constructed* F, the "iff" stated in (1) will hold.

($\leftarrow$) direction: We prove that the right hand side (rhs) of (1) implies the left, if the letter f is replaced throughout by F. The rhs is a disjunction, so we have two cases to consider [cf. 1.1.1.48(a)]. First, let $x \in \mathcal{I}$ and $y = h(x)$. Since $h = \mathcal{J} \subseteq F$, we have that $F(x, y)$ is true, that is, $y = F(x)$.

Second, consider the complicated side of OR in (1): So let, for some $R \in \mathcal{O}$, $y = g_R(x, o_1, \ldots, o_r)$, where $R(a_1, \ldots, a_r, x)$ and $o_i = F(a_i)$, for $i = 1, \ldots, r$.

By (2), $\widehat{R}\big(\langle a_1, o_1\rangle, \ldots, \langle a_r, o_r\rangle, \langle x, y\rangle\big)$, thus—$F$ being closed under all the rules in $\widehat{R}$—$\langle x, y\rangle \in F$; for short, $y = F(x)$.

($\rightarrow$) direction Now we assume that $F(x, y)$ holds. We want to infer the right hand side (of *iff*) in (1)—with f replaced by F.

So let $y = F(x)$. There are two cases according to 1.6.0.15:

Case 1. $\langle x, y\rangle \in \mathcal{J}$. Thus (by $\mathcal{J} = h$) $y = h(x)$, and $x \in \mathcal{I}$ (definition of $\mathcal{J}$); the top case of (1).

Case 2. Suppose next that $\langle x, y\rangle \in F$ because, for some $\widehat{Q} \in \mathcal{T}$, the following hold (see (2)):

(a) $\widehat{Q}(\langle a_1, o_1\rangle, \ldots, \langle a_r, o_r\rangle, \langle x, y\rangle)$

(b) $Q(a_1, \ldots, a_r, x)$

(c) $y = g_Q(x, o_1, \ldots, o_r)$

(d) All of $\langle a_1, o_1\rangle, \ldots, \langle a_r, o_r\rangle$ are in F

By (d), $o_i = F(a_i)$, for $i = 1, \ldots, r$. But then the conjunction of the foregoing observation with (b) and (c) is the right hand side of the OR; as needed. *End of proof: Claim 2.*

Uniqueness of F. Let the function K also satisfy (1). We show by induction over $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ that

$$\text{For all } x \in \mathrm{Cl}(\mathcal{I}, \mathcal{O}) \text{ and all } y \in B, \quad y = F(x) \textbf{ iff } y = K(x) \tag{4}$$

($\rightarrow$) Let $x \in \mathcal{I}$, and $y = F(x)$. By lack of ambiguity, x has no i.p. Thus, $y = h(x)$. But then, $y = K(x)$, since K satisfies (1).

Let now $Q(a_1, \ldots, a_r, x)$ and $y = F(x)$. By (1), there are (unique, as we now know) $o_1, \ldots, o_r$ such that $o_i = F(a_i)$, for $i = 1, \ldots, r$, and $y = g_Q(x, o_1, \ldots, o_r)$. By the I.H., $o_i = K(a_i)$. As K satisfies (1), $y = K(x)$.

($\leftarrow$) The roles of the letters F and K in the above argument being symmetric, we need say no more. □

The above formulation with the so-called "graphs" of the utilized functions—a term applying to the relation $y = f(x)$[47]—rather than writing, say,

$$f(x) = \begin{cases} h(x) & \textbf{if } x \in \mathcal{I} \\ g_Q(x, f(a_1), \ldots, f(a_r)) & \textbf{if } Q(a_1, \ldots, a_r, x) \text{ holds,} \end{cases}$$

appears to be unnecessarily cumbersome. Cumbersome, yes, but not "unnecessarily":

In the used formulation, by keeping an eye on both the input and output sides at once, we took care of the partial function case (*total* or *not*) without having to worry about points of undefinition of the defined function or to use Kleene equality. In fact, using "=" above is incorrect in the nontotal case.

1.7.0.26 Example. Referring back to Example 1.7.0.24, we see that the defined by recursion g exists (and is, of course, unique): It is defined over the set $\mathbb{N} = \mathrm{Cl}(\{0\}, \{S\})$, where I denoted by S the successor rule:

$$S(x, x+1), \text{ for all } x \text{ in } \mathbb{N}$$

The rule is clearly unambiguous, so Theorem 1.7.0.25 applies. □

1.7.0.27 Example. Fix $n > 0$ from $\mathbb{N}$ and consider the rule R below

$$R(\langle x, \vec{y}_n\rangle, \langle x+1, \vec{y}_n\rangle), \text{ for all } x, y_i \text{ in } \mathbb{N}$$

and form $\mathbb{N}_n = \mathrm{Cl}\big(\mathcal{I}, \{R\}\big)$, where $\mathcal{I} = \{\langle 0, \vec{y}_n\rangle : \text{ for all } y_i \text{ in } \mathbb{N}\}$.

The rule R is clearly unambiguous: every $\langle x, y_n\rangle$ is either initial or has the unique i.p. $\langle x-1, y_n\rangle$. Thus Theorem 1.7.0.25 applies—$\mathbb{N}_n = \mathbb{N}^{n+1}$, of course—and enables recursive definitions like the following, based on two given functions h and g from $\mathbb{N}^n$ and $\mathbb{N}^{n+1} \times B$ respectively to some set B, to produce unique f-solutions.

$$f(x, \vec{y}_n) \simeq \begin{cases} h(\vec{y}_n) & \textbf{if } x = 0 \\ g(x, \vec{y}_n, f(x-1, \vec{y}_n)) & \textbf{otherwise} \end{cases} \tag{1}$$

As is usual, we listed the arguments, e.g., in f, in their proper order, however omitting the $\langle \ldots \rangle$ brackets.

The above recurrence is the *primitive recursion schema* of Kleene, and it will play quite an active role in the next chapter. It is customary to write the schema in this form:

$$\begin{aligned} f(0, \vec{y}_n) &\simeq h(\vec{y}_n) \\ f(x+1, \vec{y}_n) &\simeq g'(x, \vec{y}_n, f(x, \vec{y}_n)) \end{aligned}$$

[47] To plot $f(x)$ you plot the pairs (x, y) such that $y = f(x)$; hence the terminology "graph".

where g' is g, but modified to accept input x rather than $x+1$ in the first argument slot.

Incidentally, the recursion given at the very opening of this section—taking there $B = \mathbb{N}$—fits the primitive recursion schema above, so the function it defines exists and is unique. The reader will immediately recognise that the function defined there is multiplication, $n \times m$. □

1.7.0.28 Exercise. Prove that if h and g are total, then so is f defined by (1) above, so we can replace $\simeq$ by $=$. □

1.7.0.29 Example. What about a recursion like this, where we still take our inputs (of f) from $\mathbb{N}$?

$$f(x,\vec{y}_n) \simeq \begin{cases} h(\vec{y}_n) & \textbf{if } x = 0 \\ g\Big(x,\vec{y}_n,\big\{f(0,\vec{y}_n),\ldots,f(x-2,\vec{y}_n),f(x-1,\vec{y}_n)\big\}\Big) & \textbf{otherwise} \end{cases}$$

Before we answer this, a few comments on intentions, and notation. We intend that the value $f(x,\vec{y}_n)$ is computed based on our knowledge of the *entire history of values* of f—or *course-of-values* [Kleene (1952)]—at the set of all previous "points"

$$\Big\{\langle 0,\vec{y}_n\rangle,\langle 1,\vec{y}_n\rangle,\langle x-1,\vec{y}_n\rangle\Big\} \tag{1}$$

As we can have no functions of a *variable number of arguments*, we have tentatively grouped the entire history into a single *set*-argument. It turns out that it is more profitable to use a set of *pairs* of inputs *and* outputs (of f) rather than just outputs in the recursive call embedded in g above, since such pairs can naturally handle nontotal functions—the pair $\langle a, f(a)\rangle$ is listed iff $f(a)\downarrow$.

$$\Big\{\big\langle\langle 0,\vec{y}_n\rangle,f(0,\vec{y}_n)\big\rangle,\ldots,\big\langle\langle x-2,\vec{y}_n\rangle,f(x-2,\vec{y}_n)\big\rangle,\big\langle\langle x-1,\vec{y}_n\rangle,f(x-1,\vec{y}_n)\big\rangle\Big\}$$

$$\begin{aligned} f(0,\vec{y}_n) &\simeq h(\vec{y}_n) \\ f(x+1,\vec{y}_n) &\simeq g'\Big(x,\vec{y}_n,\Big\{\big\langle\langle 0,\vec{y}_n\rangle,f(0,\vec{y}_n)\big\rangle,\ldots,\big\langle\langle x-1,\vec{y}_n\rangle,f(x-1,\vec{y}_n)\big\rangle\Big\}\Big) \end{aligned}$$

The switch to g' from g reflects the modification to the x-argument and to the set-argument.

Now, if we call the set (1) "$S_{x-1,\vec{y}_n}$" for the sake of convenience, we may rewrite the above recurrence more correctly, and without "...", as

$$f(0,\vec{y}_n) \simeq h(\vec{y}_n) \tag{2}$$

$$f(x+1,\vec{y}_n) \simeq g'\Big(x,\vec{y}_n,f \restriction S_{x,\vec{y}_n}\Big) \tag{3}$$

Here is now why the recurrence (2)–(3) has a unique f-solution: Let us write $H(x, \vec{y}_n)$ as an abbreviation of $f \restriction S_{x,\vec{y}_n}$. We can then have—using (2)–(3)—a (simple) primitive recursion (as in 1.7.0.27) for H:

$$H(0, \vec{y}_n) \simeq \left\{ \langle \langle 0, \vec{y}_n \rangle, h(\vec{y}_n) \rangle \right\} \tag{4}$$

$$H(x+1, \vec{y}_n) \simeq H(x, \vec{y}_n) \cup \left\{ \left\langle \langle x+1, \vec{y}_n \rangle, g'\left(x, \vec{y}_n, H(x, \vec{y}_n)\right) \right\rangle \right\} \tag{5}$$

A unique H exists that satisfies (4)–(5), by 1.7.0.27. But $f(x, \vec{y}_n) \simeq H(x, \vec{y}_n)\left(x, \vec{y}_n\right)$ for all $x, \vec{y}_n$; so f exists and is unique.[48]

> Given that $H(x, \vec{y}_n) = f \restriction S_{x,\vec{y}_n}$—a single-valued table of tuples—we see that evaluating $H(x, \vec{y}_n)$ at $\langle i, \vec{y}_n \rangle$, for $i = 0, 1, \ldots, x$, we end up with the output (if it exists) of f at input $\langle i, \vec{y}_n \rangle$. That is, the expression "$H(x, \vec{y}_n)(x, \vec{y}_n)$" above makes sense, and, when defined, teases out $f(x, \vec{y}_n)$.

If we work strictly within arithmetic—that is, we allow no set arguments, in particular—then one neat way to deal with a sequence of numbers is to *code them by a single number*. Applying this trick reduces once again the original recursion to the standard schema of Example 1.7.0.27. This approach has additional important side benefits and we will revisit it in the next chapter. □

1.7.0.30 Example. Let $A = \{1, 2\}$. In this example we consider only functions with inputs from A^* and outputs in A^*. Suppose that h, g_1 and g_2 are such given functions of n for the first and $n+2$ variables for the other two. The recursion (for fixed $n > 0$)

$$\begin{aligned} f(\epsilon, \vec{y}_n) &\simeq h(\vec{y}_n) \\ f(x * 1, \vec{y}_n) &\simeq g_1(x, \vec{y}_n, f(x, \vec{y}_n)) \\ f(x * 2, \vec{y}_n) &\simeq g_2(x, \vec{y}_n, f(x, \vec{y}_n)) \end{aligned}$$

is called *right primitive recursion on notation*—"right" since we change x by concatenating a 1 or 2 to its right; "on notation" since we are thinking in terms of the notation rather than value of x when we increment it.

Given the h and the g_i there is a unique f that satisfies the three equalities above, for all $x, \vec{y}_n$. To apply Theorem 1.7.0.25 we define A_n^* as $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ where $\mathcal{O}$ has two rules,

$$R_1(\langle x, \vec{y}_n \rangle, \langle x * 1, \vec{y}_n \rangle), \text{ for all } x, \vec{y}_n \text{ in } A^*$$

and

$$R_2(\langle x, \vec{y}_n \rangle, \langle x * 2, \vec{y}_n \rangle), \text{ for all } x, \vec{y}_n \text{ in } A^*$$

We define $\mathcal{I}$ as $\{\langle \epsilon, \vec{y}_n \rangle : \text{ for all } y_i \text{ in } A^*\}$.

Clearly the pair $\mathcal{I}, \mathcal{O}$ is unambiguous and 1.7.0.25 applies to the recursion on notation schema above, proving existence and uniqueness of f.

[48] The "g-function" in (5) is $G(x, \vec{y}_n, Z) \simeq Z \cup \left\{ \left\langle \langle x+1, \vec{y}_n \rangle, g'\left(x, \vec{y}_n, Z\right) \right\rangle \right\}$, where Z is a set argument. If the right field of the original h and g is a set B, then Z takes its values from $\mathcal{P}(\mathbb{N}^{n+1} \times B)$.

One can similarly utilize *left* recursion on notation, going from x to $i * x$, for $i = 1$ or $i = 2$. □

1.8 ADDITIONAL EXERCISES

1. *Let us first define:* The set of propositional formulae of, say, set theory, denoted here by **Prop**, is the smallest set such that

(1) Every Boolean variable is in **Prop** (cf. 1.1.1.26)

(2) If $\mathscr{A}$ and $\mathscr{B}$ are in **Prop**, then so are $(\neg\mathscr{A})$ and $(\mathscr{A} \circ \mathscr{B})$ —where I used $\circ$ as an abbreviation of any member of $\{\wedge, \vee, \rightarrow, \equiv\}$.

If we call **WFF** the set of all formulae of set theory as defined in 1.1.1.3, then show that **WFF** = **Prop**.

Hint. This involves two structural inductions, one each over **WFF** and **Prop**.

2. Prove the general case of proof by cases (cf. 1.1.1.48): $\mathscr{A} \rightarrow \mathscr{B}, \mathscr{C} \rightarrow \mathscr{D} \vdash \mathscr{A} \vee \mathscr{C} \rightarrow \mathscr{B} \vee \mathscr{D}$.

3. Let us prove $\vdash x = y$. By way of contradiction, let us assume $\neg x = y$ (i.e., $x \neq y$). Using substitution (1.1.1.42) we obtain $\neg x = x$ which along with axiom (v) (1.1.1.38) and tautological implication yields the contradiction $x = x \wedge \neg x = x$. Done.

Hm. There is something very wrong here! Clearly, $x = y$ is not true, hence a proof of it must be impossible (cf. soundness, p. 20). What *exactly* went wrong with our "proof"?

4. Verify that for any (formal) function $f[x]$ we have $\vdash x = y \rightarrow f[x] = f[y]$.

Hint. Start with (vi) of 1.1.1.38 taking as "$\mathscr{A}[z]$" the formula $f[x] = f[z]$.

5. Give the missing details of Example 1.1.2.14.

6. This is a useful but simple exercise! For all sets A, B, C prove:

(i)	$A \cup A = A$	and $A \cap A = A$
(ii)	$A \cup (A \cap B) = A$	and $A \cap (A \cup B) = A$
(iii)	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	and $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
(iv)	$A \subseteq B \equiv A \cup B = B$	and $A \subseteq B \equiv A \cap B = A$

7. Compute $\bigcup\{2\}$.

8. Compute $\bigcap\emptyset$.

9. Compute $\bigcap\{7\}$.

10. Use induction on $n \geq 2$ to prove that if $\langle x_1, x_2, \ldots, x_n \rangle = \langle y_1, y_2, \ldots, y_n \rangle$, then, for $i = 1, \ldots, n$, we have $x_i = y_i$.

11. Can A^n be also defined as

$$A^1 = A$$

and, for $n \geq 1$

$$A^{n+1} = A \times A^n$$

Why?

12. What is $A \times 1$? Why?

13. Prove that $A \times \emptyset = \emptyset \times A = \emptyset$.

14. Prove that $A \times B = \emptyset$ iff $A = \emptyset$ or $B = \emptyset$.

15. What is $\mathcal{P}(A)$ if A is an urelement?

16. Assume an intuitive understanding of "the set A has n elements". Prove by induction on n that 2^A has 2^n elements. This motivates the notation "2^A".

Hint. Show carefully that adding one new element to A doubles the number of its subsets.

17. Show that for any function $f : A \to B$, $f(a) \uparrow \equiv f_{\to}(\{a\}) = \emptyset$; and f is onto iff $(\forall x \in B) f_{\leftarrow}(\{x\}) \neq \emptyset$.

18. Show by an example that function composition is not commutative. That is, in general, $(gf) \neq (fg)$.

19. For any function g and sets X and Y, we have $g_{\leftarrow}(X - Y) = g_{\leftarrow}(X) - g_{\leftarrow}(Y)$.

20. Let $f : X \to Y$ be given as well as $A \subseteq Y$ and $B \subseteq Y$. Prove that

- $f_{\leftarrow}(A \cup B) = f_{\leftarrow}(A) \cup f_{\leftarrow}(B)$
- $f_{\leftarrow}(A \cap B) = f_{\leftarrow}(A) \cap f_{\leftarrow}(B)$

21. Let $f : X \to Y$ be given as well as $A \subseteq X$ and $B \subseteq X$. Prove that

- $f_{\to}(A \cup B) = f_{\to}(A) \cup f_{\to}(B)$
- $f_{\to}(A \cap B) \subseteq f_{\to}(A) \cap f_{\to}(B)$
- $B \subseteq A$ implies that $f_{\to}(A - B) \supseteq f_{\to}(A) - f_{\to}(B)$

22. Let $f : X \to Y$ be given as well as $A \subseteq X$ and $B \subseteq Y$. Prove that

- $f_{\to}\Big(f_{\leftarrow}(B)\Big) \subseteq B$
- $f_{\leftarrow}\Big(f_{\to}(A)\Big) \supseteq A$

23. Let the relation $R : \bigtimes_{i=1}^{n} A_i \to A_k$ be given by

$$R = \{\langle\langle a_1, \ldots, a_k, \ldots, a_n\rangle, a_k\rangle : a_j \in A_j, \text{for } j = 1, \ldots, n\}$$

(1) Prove that R is a total function. We call it the k-th *Cartesian projection* function of $\bigtimes_{i=1}^{n} A_i$, and often denote it by p_k^n.

(2) Prove that if $f : B \to \bigtimes_{i=1}^{n} A_i$ is a (*vector-* or *tuple-valued*) function, then we may *decompose* it into n *functions*, f_i, for $i = 1, \ldots, n$: $f_i : B \to A_i$, so that, for all $a \in \text{dom}(f)$, we have $f(a) = \langle f_1(a), \ldots, f_n(a)\rangle$.

We say that the f_i is the i-th *component* or *projection* of the tuple-valued (vector-valued) function f.

Hint. Consider $(p_i^n f)$ (i.e., $f \circ p_i^n$).

24. Prove Theorem 1.2.0.19.

25. Suppose that $f : A \to B$. Then $(\mathbf{1}_B f) = f$ and $(f\mathbf{1}_A) = f$.

26. Let $f : A \to B$ be a 1-1 correspondence. Then show that $g = f^{-1} : B \to A$ is a 1-1 correspondence as well and $(fg) = \mathbf{1}_B$ while $(gf) = \mathbf{1}_A$.

27. Consider $f : A \to B$, $g : B \to A$ and $h : B \to A$ such that $(fg) = \mathbf{1}_B$ and $(hf) = \mathbf{1}_A$ hold.

Show that f is a 1-1 correspondence and that $g = h = f^{-1}$.

Hint. Start with expanding $(h(fg)) = (h\mathbf{1}_B)$, using associativity and Exercise 25.

28. Let $f : A \to B$ be a 1-1 correspondence and $g : B \to A$ be a function for which $(gf) = \mathbf{1}_A$. Then show that $g = f^{-1}$ and therefore $(fg) = 1_B$ as well.

29. Let $f : A \to B$ be a 1-1 correspondence and $g : B \to A$ be a function for which $(fg) = \mathbf{1}_B$. Then show that $g = f^{-1}$ and therefore $(gf) = 1_A$ as well.

Exercises 27, 28 and 29 show that if an f has both a left and a right inverse, then the two are equal to f^{-1} and in fact f is a 1-1 correspondence. Moreover, a 1-1 correspondence has a unique left and right inverse, equal in each case to f^{-1}.

This unique inverse is called—for 1-1 correspondences—"the" (two-sided) inverse.

30. Prove that R is transitive iff $R^2 \subseteq R$.

31. Prove that if $A \sim B$ and $B \sim C$ then $A \sim C$.

32. Prove, for any two functions f and g, that $f = g$ (as sets of tuples) iff $(\forall x)(f(x) \simeq g(x))$.

33. Prove the claims made in Remark 1.2.0.31.

34. Let $f : A \to B$ and $g : B \to C$ be onto functions. Show that $(gf) : A \to C$ is onto.

Hint. An $h : X \to Y$ is onto iff for any $b \in Y$, the "equation" $h(x) = b$ has a solution.

35. Revisit Theorem 1.3.0.42 and give it a mathematical proof, using tools from Section 1.4. Specifically, if $A \subseteq \mathbb{N}$ is infinite, define by recursion the function f by $f(0) = \min A$ and $f(n+1) = \min(A - \{f(0), \ldots, f(n)\})$ and prove that $\mathrm{dom}(f) = \mathbb{N}$, $\mathrm{ran}(f) = A$ and f is 1-1.

36. Prove that the range of $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ given by $f(x, y) = 2^x 3^y$ is infinite.

37. Prove that there is a 1-1 correspondence that corresponds each $S \subseteq \mathbb{N}$ to χ_S.

38. Let A and B be enumerable. Then $2^A \sim 2^B$.

Hint. Let $f : \mathbb{N} \to A$ and $g : \mathbb{N} \to B$ be 1-1 correspondences. Define $F : 2^A \to 2^B$ so that $F(\emptyset) = \emptyset$ and F sends the set $\{f(i_0), f(i_1), f(i_2), f(i_3), \ldots\}$ to the set $\{g(i_0), g(i_1), g(i_2), g(i_3), \ldots\}$. Argue that F is total, 1-1, and onto.

39. Refer to 1.6.0.19. Define the simple arithmetic terms of that example differently: Let us start with the alphabet of symbols $A = \{1, 2, 3, +, \times, (,)\}$. We let $\mathcal{I} = \{1, 2, 3\}$ and $\mathcal{O}$ consist of just two *string* operations:

$$\text{from strings } X \text{ and } Y \text{ form } (X + Y) \tag{1}$$

$$\text{from strings } X \text{ and } Y \text{ form } (X \times Y) \tag{2}$$

Prove that $\mathcal{I}, \mathcal{O}$ is unambiguous and thus EV, defined as in 1.6.0.19, over $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ exists and is unique.

Toward a proof of lack of ambiguity you may want to prove a couple of lemmata:

(a) Every member of $\mathrm{Cl}(\mathcal{I}, \mathcal{O})$ has an equal number of left and right brackets.

(b) Every *proper* non-empty string *prefix* of an $A \in \mathrm{Cl}(\mathcal{I}, \mathcal{O})$ has an excess of left brackets.

(c) Every $A \in \mathrm{Cl}(\mathcal{I}, \mathcal{O})$ has unique i.p. and $1, 2, 3$ have no i.p.

40. Prove, as Euclid did, that every natural number $n > 1$ is a product of primes in a unique way, except for permutation of factors.

Hint. Use strong induction and 1.4.1.3.

41. Prove that if $R : A \to A$ is reflexive and also satisfies, for all x, y and z,

$$xRy \wedge xRz \to yRz$$

then it is also symmetric and transitive, hence an equivalence relation.

42. Prove, by induction on n, that for any relation R on a set A we have

(1) $R^m \circ R^n = R^{m+n}$

(2) $(R^m)^n = R^{mn}$

43. Suppose that R is on a finite set of n elements. Prove that $R^+ = \bigcup_{i=1}^{n} R^i$.

Hint. Cf. 1.6.0.22. Prove the redundancy of all terms beyond R^n in this case.

44. Suppose that R, defined on a finite set of n elements, is reflexive. Prove that $R^+ = R^{n-1}$.

Hint. Prove the redundancy of all terms but R^{n-1} in this case.

45. Let $m > 1$ be an integer. Prove that any integer $n > 0$ can be *uniquely* written as $n = mq + r$, where $0 < r \leq m$.

Hint. Note the inequalities! Either imitate the *proof* given in 1.4.3.1, or base a proof on the *result* of 1.4.3.1.

46. **(m-ary notation.)** Prove that every integer $n \geq 0$ has a unique representation as

$$n = d_r m^r + d_{r-1} m^{r-1} + d_{r-2} m^{r-2} + \cdots + d_0 \tag{1}$$

where $0 \leq d_i < m$ for all $i = 0, \ldots, r$. (1) is called *the* m-ary notation of n, and the d_i are the m-ary digits.

47. **(m-adic notation.)** Prove that every integer $n > 0$ has a unique representation as

$$n = d_r m^r + d_{r-1} m^{r-1} + d_{r-2} m^{r-2} + \cdots + d_0 \tag{2}$$

where $0 < d_i \leq m$ for all $i = 0, \ldots, r$. (2) is called *the* m-adic notation of n [cf. Smullyan (1961); Bennett (1962)], and the d_i are the m-adic digits.

48. Prove that for any set X, we have $X \not\sim 2^X$.