Foundations of Mathematics

Martin Ward

March 23, 2015

Contents

1. DEDUCTIVE SYSTEMS

A Formal languages

In this first section I introduce a number of ideas, for the most part without defining them properly. The purpose is to give a rough idea of the general direction we will be heading in. Better definitions will be given later.

A.1 The idea of a formal language

Let us suppose we have decided to develop a self-contained theory of the natural numbers, treating them with full mathematical rigour. Our writings would have informal discursive portions, which would be written in English (or whatever language we choose) peppered with some mathematical technical terms, and there would be highly formal portions, such as equations, proofs and so on, which would not look like a natural language at all. To be entirely rigorous, the full logical development of the theory should be contained in the formally written part, which would consist for the most part of definitions and proofs. The discursive portions would be limited to discussion of motivation, pointing out interesting connections, history and so on and could be dispensed with without affecting the logical structure of the theory. Our focus will be on the formal language (which is all that is logically necessary).

To treat the natural numbers in this formal way, we would need some symbols: some letters to refer to variables, some numeric digits such as 0 and 1, and some logical and mathematical symbols such as $+$, $=$, $\leq$, $\forall$ and so on; perhaps quite a short list. (We will have good reason to do exactly this in Chapter **4** and the list is indeed short.)

▶ Ch.4

The statements we make in our formal language will consist of strings of these symbols. Some of the strings will be just plain rubbish, such as

$$)\forall + (=== .$$

Of course we hope we will not write such strings down. The kind of strings we will be writing are ones which are meaningful, which we will call *expressions*, for instance

$$(\forall x)\,(\,x + 1 > x\,)$$
$$(\forall x)\,(\,x + y > x\,)$$
$$2 + 2 = 5$$

While an expression must make sense, it need not be true. The first example above is true, the last one is false and the middle one occupies an intermediate position — its truth or falsity depends upon the value of y.

We expect that it should be a fairly simple matter to decide whether a string is an expression or not. In most areas of mathematics you only have to glance at an equation to see if it is

well-formed or not (assuming you are already familiar with the notation used in that area). You might have trouble with a very complicated equation which extended over several lines, and then you could engage in a more careful procedure — checking that parentheses match and so on. In short, one has an algorithm (a straightforward computational procedure) for determining whether a string is an expression or not; one hopes it is simple enough that it can be applied to short expressions very easily, even subconsciously.

Putting this together, we have a *formal language*. In general, a formal language consists of

- A defined set of *symbols*, which we will call its *alphabet*, even though it is likely to contain symbols other than ordinary letters.

  We will be interested in *strings*, by which we mean strings of members of the alphabet.
- A defined set of *expressions*, which is a subset of the set of all strings. (Expressions are sometimes called *well-formed formulas* or just *wffs*.)
- An algorithm which will decide whether a string is an expression or not.

Returning to our number theory example, we are not finished yet of course: our primary interest is in the set of expressions which are in fact true. We will call the set of true expressions the *theory* and we expect it to be well-defined in some way. We may, for instance, define it axiomatically, starting with Peano's Axioms and defining the theory as consisting of those expressions which can be derived from the chosen axioms by valid proofs. Alternatively, we might know all about the natural numbers already, from some other area of Mathematics perhaps, and simply define our theory as the set of expressions which say something which is in fact true about them.

We now have a *formal system*. In general, a formal system (in the sense we will use this phrase in these notes) consists of a formal language together with a well-defined subset of its expressions called its *theory*.

The theory may be defined in various ways, of which there are two particularly important ones, already briefly mentioned above.

- Firstly the language may be used to describe some object (for example, our language for the natural numbers would describe the set $\mathbb{N}$ of natural numbers and its structure) or any one of a class of objects (for example, a language for group theory might describe any group and its structure). Assuming we already know all about that structure or class of structures (whatever that means), it would be natural to define the theory to be the set of expressions which are true of that object or every one of that class of objects. In this case we say that the theory is defined *semantically*.
- Secondly, the theory might be described entirely internally, in terms of the set of expressions alone with no reference to any external structure which the language is supposed to describe. In this case we say that the theory is defined *syntactically*.

  As an important special case, we define a particular set of expressions which we call *axioms* together with a set of rules for deducing new expressions from old ones. We then define the theory to be the set of expressions which can be produced from the given axioms by the given rules. In this case we have a *deductive system*.

In these notes we will primarily be interested in deductive systems.

It is not necessary for a language to be mathematically oriented. It is possible, for instance, to define a language in which the strings specify four-part musical pieces and the theory consists of those pieces which obey the rules of classical (common practice) harmony. For perhaps a more obvious example, the language could be a computer language, C for instance; in this case the expressions would be programs obeying the syntax of the language and we could choose, as our theory, the set of programs which do not crash (in some sense which we would have to specify carefully). Then an example of a theorem would be a bug-free program, for instance the well-known "Hello world". The expressions in this language (= programs) tend to be long and very complicated and the algorithm to distinguish expressions from non-expressions correspondingly intricate: this is (part of) what a compiler does.

The interplay between semantics and syntactics is particularly interesting. For example, we might have a theory defined semantically which describes an important object or class of objects, and be interested in providing a deductive basis for the theory — or indeed in discovering whether it is possible to provide such a basis at all. (Can everything true about the Natural Numbers be deduced from Peano's Axioms or do we need a better basis? If so, is it possible to provide such a better basis?) Alternatively, we may have a deductive system and wish to elucidate what kinds of objects it describes. (Are there structures out there which are not isomorphic to the Natural Numbers but which are nevertheless described truly by the theory derived from Peano's Axioms?)

As stated above, it is part of the idea of a language that it should be possible to decide algorithmically whether a string is an expression or not (in mathematical jargon, the set of expressions is *decidable* or *recursive*). It is normally expected that this algorithm be fairly simple. On the other hand, there is no guarantee that any algorithmic method exists to decide whether an expression is a member of the theory or not. Indeed, there are theories (mathematics itself included) in which no such algorithm exists.

Some examples of theories:

- Natural numbers (peano's axioms).
- Finite set theory.
- Real numbers.
- "All" of mathematics.
- Simple formal systems such as the one described in the next section.

When we come to investigating a particular language, for instance that for Mathematics, we will define it completely formally. We do this because we are interested in proving things about mathematical proofs, provability, and theories in general, some of them subtle and some surprising; we need a firm basis to work from. This means that we must idealise our language somewhat — it must differ to some extent from common usage. Given that we are going to do this, we may as well idealise it in a way which is convenient for our present purposes. It should be understood however that, whatever we do, anything correctly stated in ordinary mathematical discourse should be translatable into our formalised version of the

language and vice versa. Having granted ourselves this freedom, we will define the formal language as simply as possible. This will mean that it is (relatively) easy to prove things about it. On the other hand, as we will see, it will be very difficult to read and most impracticable to use. We will usually employ a number of abbreviations to make it easier to understand — thus creating a "user friendly", *semi-formal* version of the language.

Many languages embed logic in some sense. There are many different logics and many ways they could be incorporated into a language, however two forms of logic stand out as most important: *Sentential* (or *Propositional*) Logic and *Predicate* (or *First Order*) Logic. Sentential Logic is concerned with *propositions* (or *sentences* or *statements*), that is, expressions which are either true or false (but not both) and the logical relations between them, the most common being negation, conjunction, disjunction, implication and equivalence. Predicate Logic includes all of this and adds the quantifiers ("For all x ..." and "there exists x such that ..."). That, of course, was a rough description, not a definition.

Sentential and Predicate Logic will be described in detail in the next two chapters, however to continue these introductory remarks, let us assume that we now know what Predicate Logic is. There are several ways we could embed it into our language. For a start, there is a choice of symbols: we do not need all of $\neg$ $\wedge$ $\vee$ $\Rightarrow$ $\Leftrightarrow$ $\forall$ $\exists$, because some of these symbols can be defined in terms of the others. When we have chosen our symbols and the rules for forming expressions, we have some choices about the theory. We have (I hope) a pretty good idea as to what the theory should consist of, but how are we to define it? — semantically or deductively, and if deductively, what will we take as our axioms and deductive rules? To a large extent, these choices are simply a matter of convenience: any definition we choose will be acceptable provided it yields a language and theory which is equivalent to "standard" Predicate Logic ("equivalent" in some appropriate sense — see Section **2**.I for a careful description of this). In these notes we adopt a particularly simple and fairly common definition.

► **2**.I

Suppose now we have some language in which we are interested (Mathematics, Natural Numbers only, Common Practice Harmony in music, the computer language C or whatever). Here are some fundamental questions we might ask about it:

- Is the theory *decidable*? That is, is there an algorithm which will decide whether a given expression is a member of the theory or not?
- If the theory is defined semantically, is it *axiomatisable*? In other words, can we redefine it as a deductive system?
- If it is defined deductively, what sort of interpretations does it have, perhaps quite different from whatever led us to think it up in the first place?

Those questions can be asked whether the language has logic embedded in it or not. Now suppose that the language contains (at least) sentential logic.

- Is the theory *consistent*? A theory is *inconsistent* if it contains two expressions of the forms P and NOT–P (in whatever symbology the language employs). In an inconsistent theory, it would then follow from the rules of Sentential Logic that every

expression is a member of the theory (and therefore so would the negation of every expression), and so the whole thing becomes trivial. Indeed, worse than that, a theory which contained such expressions would also contain (one meaning) P AND NOT P, and so would violate the spirit of sentential logic anyway. In short, we really do not want any of our theories to be inconsistent.

- Is the theory *complete*? A theory is complete if, given any sentence P, either P or NOT–P is a member of the theory. Since a sentence usually represents some proposition which is either true or false, one would hope that the theory was strong enough to say which of these was the case: completeness means that this is so. (I have not defined what a *sentence* is yet. In Sentential Logic, every expression is a sentence – hence its name; in Predicate Logic things are a bit more complicated. Whether an expression is a sentence or not depends on how any variables which occur in it are treated there. The idea is defined properly in Chapter **3**.) ▶ Ch.**3**

- Is the theory *categorical*? A theory is categorical if all interpretations of it are isomorphic. If we have an axiomatisation of the Natural Numbers, for instance, we could imagine building several different models of this theory, but we would hope that the differences between the different versions would be trivial, that the axiomatisation had indeed captured *all* the structure of the Natural Numbers. (In Sections **7**.B.10 and **7**.C.3 we will see, alarmingly, that this is not so). On the other hand, we could axiomatise Classical Harmony for music in such a way that it would yield normal harmony in the standard twelve-note scale and something quite different in a fifteen-note scale (with the same axioms) — this would be not categorical. By the second example I am hoping to suggest that failure to be categorical, in some cases, may be a good thing. ▶ **7**.B.10 ▶ **7**.C.3

A.2 Example: A "toy" formal system

In this system there are three symbols: **l** **a** **b**

All strings are expressions. (This is unusual amongst languages, but it certainly makes it easy to decide whether a string is an expression or not.)

There is only one axiom, namely **l**

There are four rules:

(1) If an expression contains the symbol **l**, then it may be removed (provided the result is not empty). Conversely an **l** may be inserted anywhere.

$$x\mathbf{l}y \quad \leftrightarrow \quad xy \ .$$

(2) The string **aaa** may be replaced by **l** anywhere, and vice versa. That is

$$x\mathbf{aaa}y \quad \leftrightarrow \quad x\mathbf{l}y \ .$$

(3) And the same for **bbb**:

$$x\mathbf{bbb}y \quad \leftrightarrow \quad x\mathbf{l}y \ .$$

(4) The string **abab** may be replaced by **bbaa** anywhere, and vice versa:

$$x\mathbf{abab}y \quad \leftrightarrow \quad x\mathbf{bbaa}y \ .$$

(In these rules, x and y stand for any subexpressions.)

An expression is part of the theory if it can be "deduced" from the axiom by a sequence of none or more of the rules. For example, we could have a sequence

I	$\rightarrow$ **aaa**	by Rule 2
	$\rightarrow$ **Iaaa**	by Rule 1
	$\rightarrow$ **bbbaaa**	by Rule 3
	$\rightarrow$ **bababa**	by Rule 4

which would "prove" that **bababa** is a member of the theory, i.e. a "theorem".

In discussing this theory, it is very tempting to use notations such as $\mathbf{a}^3$ and $(\mathbf{ab})^2$. There is no harm in doing this, so long as one remembers that such notation is not part of the formal language we are discussing, but rather just a convenient abbreviation to use when discussing it. We say that it is not part of the formal language, but part of a *semi-formal* version of it.

Using these abbreviations, the rules become a bit easier to read:

$x\mathbf{I}y$	$\leftrightarrow$	xy	Rule 1
$x\mathbf{a}^3y$	$\leftrightarrow$	$x\mathbf{I}y$	Rule 2
$x\mathbf{b}^3y$	$\leftrightarrow$	$x\mathbf{I}y$	Rule 3
$x(\mathbf{ab})^2y$	$\leftrightarrow$	$x\mathbf{b}^2\mathbf{a}^2y$	Rule 4

and the example proof:

$$\mathbf{I} \ \rightarrow \ \mathbf{a}^3 \ \rightarrow \ \mathbf{I}\mathbf{a}^3 \ \rightarrow \ \mathbf{b}^3\mathbf{a}^3 \ \rightarrow \ (\mathbf{ba})^3$$

Here are some questions to try. I believe that they are in increasing order of difficulty.

(Q1) Is $(\mathbf{ba}^2\mathbf{b}^2\mathbf{aba})^4$ a theorem?

(Q2) Is $(\mathbf{abab}^2)^3$ a theorem?

(Q3) Is $(\mathbf{ab}^2)^3$ a theorem?

The point of these questions is to illustrate that, even with quite a simple theory, the decidability question may not be at all straightforward. And this leads to the more general ...

(Q4) Is this theory decidable or not? In other words, is there an algorithm (a routine computational process) which will decide, for any expression of the language, whether it is a member of the theory or not?

A.3 Strings

In setting up a language, we start with a set of symbols. Let us assume for now that we have decided on these.

We now make strings of symbols, e.g. $2+2=4$ and $xy\forall+-$. Nonsensical sequences of symbols are included. To facilitate discussion, we also include the *empty sequence*, which I will denote by the name $\boldsymbol{e}$.

> Pause for nitpicking. We cannot write the empty string itself, except I suppose to display a blank space on the paper. One could argue whether there is such a thing as the empty string at all. It suffices to say that anything we will say below which mentions the empty string explicitly or implicily can also be said without it; we are simply using the idea as a notational or terminological convenience. In any case, the symbol $\boldsymbol{e}$ which I use to represent the empty string is not the empty string itself but a name for it, and so is not part of the formal language.

We note that two strings can be concatenated, i.e. joined end to end. We can concatenate abc and pqa to get abcpqa. We could concatenate the and cat to get thecat. This turns our set of expressions into an algebraic system, consisting of a set $\mathcal{S}$ of strings, with two operations

$$\begin{array}{cc} \text{the empty string } e & \text{(nullary)} \\ \text{concatenation} & \text{(binary)} \end{array}$$

It is easy to see that the empty string acts as an identity and that concatenation is associative, i.e. that the following laws hold:

$$\begin{array}{ll} u\boldsymbol{e}=u \quad \text{and } \boldsymbol{e}u=u & \text{for any string } u, \\ u(vw)=(uv)w & \text{for any strings } u,\ v \text{ and } w. \end{array}$$

This means that $\mathcal{S}$ is a monoid, that is, a semigroup with identity. This particular semigroup has some very special properties.

> (In normal algebraic terminology, we would refer to $\boldsymbol{e}$ as the *identity* and the operation as *multiplication*. In discussing the monoid of strings from an algebraic point of view, it is difficult to avoid shifting back and forth between the two terminologies.)

In an arbitrary monoid, an element u is *decomposable* if it is of the form $u=xy$ where neither x nor y is the identity; otherwise it is *indecomposable*. Obviously, the indecomposable elements of $\mathcal{S}$ are just the symbols.

A monoid has the *unique decomposition property* if every string can be written as a product of indecomposable elements in exactly one way. Our monoid of strings has this property. This allows us to define the *length* of a string as the number of symbols (=indecomposables) it contains. If

$$u=s_1s_2\dots s_k, \quad \text{where } s_1,s_2,\dots,s_k \text{ are symbols (indecomposables)}$$

then the length of u is k.

(Note that, to cover the case of the empty string, we interpret this to say that the empty string is written as the product of no symbols, and so its length is 0). We observe that

$$\text{length}(\boldsymbol{e}) = 0$$

and

$$\text{length}(uv) = \text{length}(u) + \text{length}(v) \qquad \text{for any strings } u \text{ and } v.$$

This allows us to define and prove things about strings by induction over their length.

The monoid $\mathcal{S}$ is *cancellative*, that is it has the properties that

$$\text{If } \; ux = uy \; \text{ then } \; x = y.$$
$$\text{If } \; xu = yu \; \text{ then } \; x = y.$$

A.4 Definition

(i) A string u is *part* of a string v if there are strings x and y such that $v = xuy$.

(Note that either x or y or both may be empty here, so in particular, any string is part of itself. Also the empty string is part of every string.)

(ii) An *occurrence* of the string u in the string v is a triple (x, u, y) such that $v = xuy$.

Suppose, further, that there is an occurrence of v in w, given by (x', v, y') such that $w = x'vy'$. Then clearly we have an occurrence of u in w, namely $(x'x, u, yy')$. In this case we say that this occurrence of u in u is a part of the occurrence (x', v, y') of v in w.

B Deductive systems

B.1 Introduction

A deductive system is a formalisation of the idea of setting up a theory by giving axioms and prescribed deductive rules. The theory may or may not include logic; if it does, it may contain any one of a number of possible logics. A deductive system is usually set up in an effort to build a theory of some known body of facts, but it is quite possible to create one quite abstractly. The simple **I-a-b** system, described above, is a deductive system which does not include any logic and does not appear to have any obvious interpretation. Nevertheless it is not useless: it has already served as an example that decidability may well be a difficult problem.

To define a deductive system, first of all we need a language, **L** say. The language does not come already equipped with a theory: we are going to set that up.

Next we specify (or are given) a set of *axioms*. This is simply any chosen set of expressions. It is normally assumed that the set of axioms must be decidable and certainly for any useful deductive theory this will be so. However very occasionally we may want to consider a theory in which the set of axioms is not (or need not be) decidable, so we do not want to make it a hard and fast rule. In the rest of these notes, you can assume that all sets of axioms are decidable unless I say otherwise — however it is probably a good thing, each time a new system is introduced, to look at the axioms and ask whether the set is decidable or not. (If we have only a finite set of axioms, that is automatically decidable. However we may well want the set of axioms to be infinite, in which case the decidability criterion is not quite so trivial.)

Next we specify (or are given) a set of *deductive rules*. Each of these is of the form: from a certain finite set of expressions, $A_1, A_2, \ldots, A_r$ say, one can deduce another given expression B. (More precisely, the rule simply consists of the specified expressions. The phrase "one can deduce" is, so far, a meaningless convenience. We will give it a meaning in the next paragraph or so.) Such a rule is often written in the convenient form

$$\frac{A_1, A_2, \ldots, A_r}{B} \tag{1.1}$$

Here the expressions $A_1, A_2, \ldots, A_r$ are called the *premises* and B the *conclusion* of the rule. Once again, there may be an infinite number of rules but, as with the axioms, the set of rules will normally be decidable, for the same reason.

Once we have specified the axioms and the deductive rules, the complete deductive system is specified. The theory is then built up from these as follows. First we have the idea of a *proof*. A proof is a finite sequence of expressions,

$$L_1, L_2, \ldots, L_n \qquad \text{say}$$

(which we call the *lines* of the proof) such that every one of the individual expressions L_i is either an axiom, or follows from earlier members of the sequence by one of the deductive rules.

Is it obvious what it means to say that, in this sequence, the member L_i follows from earlier members by the rule (1.1)? It means that, among the preceding members $L_1, L_2, \ldots, L_{i-1}$ of

the sequence, all the expressions $A_1, A_2, \ldots, A_r$ occur, though not necessarily in that order, and L_i is the same as B.

The definition of a proof includes sequences L_1 of length 1. In this case L_1 must be an axiom.

Now we can say what a *theorem* of this system is. It is any expression which occurs as the final member of any proof. The *theory* for this system is defined to be the set of theorems.

There are several simple facts about proofs that should be pointed out here. Firstly, the remark above about sequences of length 1 tells us that all the axioms are theorems. Secondly, note that we could have defined a theorem alternatively as any expression which occurs *anywhere* within any proof. This follows from the obvious fact that , if $L_1, L_2, \ldots, L_n$ is a proof and $1 \leq m \leq n$, then $L_1, L_2, \ldots, L_m$ is also a proof. Thirdly, if an expression appears in a proof, it may always be repeated at a later point, since whatever justified its use the first time will still justify it later on. And finally, it is possible to think of an axiom as a special kind of rule: one with a conclusion and no premises ($r = 0$ in (1.1)), but to do is seldom helpful.

B.2 Schemata

Returning to our simple $\mathbf{I} - \mathbf{a} - \mathbf{b}$ system, we have a single axiom, $\mathbf{I}$. Consider the second rule, which I wrote as $x\mathbf{a}^3y \leftrightarrow x\mathbf{I}y$. This is of course just a convenient shorthand for two rules, $x\mathbf{a}^3y \rightarrow x\mathbf{I}y$ and $x\mathbf{I}y \rightarrow x\mathbf{a}^3y$. According to our latest notation, we should write these

$$\frac{x\mathbf{a}^3y}{x\mathbf{I}y} \qquad \text{and} \qquad \frac{x\mathbf{I}y}{x\mathbf{a}^3y} \ .$$

But this does not exactly fit the definition of a deductive rule given above: the premise and conclusion are not expressions. They represent, rather, forms of expressions. Must we extend the definition above to include such rules?

The solution is to regard this, not as a single rule, but rather as a recipe for producing a whole set of rules, one for every possible pair of "values" of x and y. More precisely, substitution of any pair of expressions for x and y (possibly empty and same substitutions top and bottom) in the recipe above produces a rule. Such a recipe is called a *schema* and the individual rules which it defines are called *instances* of the schema. It is not a new kind of deductive rule, but simply a convenient way of specifying an infinite set of genuine deductive rules.

This is a *deductive rule schema*. Shortly we will see that an *axiom schema* is also a useful thing.

> (By the way, the plural of "schema" is "schemata", if you want to be pedantic. However "schemas" will do.)

It is not difficult to see that an infinite number of deductive rules is essential for an interesting theory. Observe that the only things that can turn up as expressions in proofs are axioms and the conclusions of rules. Therefore, if a deductive system had only a finite number of deductive rules (I refer to genuine rules, not schemata now), then the theory would consist

of only the axioms and a subset of this finite set of conclusions. In this case, we might as well add those conclusions to the axioms in the first place, and have a theory which has been specified simply by writing down all the theorems, and no deduction necessary.

B.3 Decidability again

It has been mentioned above that, in a deductive theory, the sets of axioms and rules are normally decidable. In such a system it follows that proofs are also decidable: it is easy to construct an algorithm which will recognise whether any given sequence of expressions is a proof or not. This does not mean that the theory itself is decidable: there is no guarantee that an algorithm exists which will decide whether or not a proof exists for any given expression.

Here is an interesting undertaking.

(i) It is not hard to make an algorithm which will list all the strings. Suppose first that there is only a finite number of symbols. To make the list, simply start with the empty string, then list all the strings of length 1 in dictionary order, followed by all the strings of length 2, followed by all the strings of length 3 and so on. It is not hard to extend this method to cover the infinite number of variables case.

(ii) Next, let us perform this algorithm, but as we go, apply the given method of deciding whether strings are expressions or not, and so throw away every string which is *not* an expression. We now have an algorithmic way of producing a list of all expressions.

(iii) Now go back to step (i), but change it in the obvious way to list all *finite sequences* of strings. Then apply the trick of step (ii) to turn it into a method of listing all finite sequences of expressions.

(iv) Tune up the last algorithm one further stage by testing each sequence of expressions, as it is produced, to see if it is a proof or not, and throwing away all non-proofs. We now have an algorithm which will list all proofs.

(v) Now use this algorithm to produce proofs, but as each one is produced, select its last expression and discard the rest. We have an algorithm which will list *all* the theorems.

How then can the set of theorems possibly be undecidable?

(A detailed answer to this important question will be given in Chapter **8**.) ► Ch.8

B.4 Deductions

We now define what it means to say "B can be deduced from H". Informally this means that, if we allow ourselves to assume H as a hypothesis, then we can construct a proof of B. We will express this in symbols

$$H \vdash B .$$

More generally, we define what it means to say that B can be deduced from several hypotheses $H_1, H_2, \ldots, H_k$; in symbols

$$H_1, H_2, \ldots, H_k \vdash B .$$

This means that there is a *proof* leading from the *hypotheses* $H_1, H_2, \ldots, H_k$ to the conclu-
▶ B.1 sion B; this is just like a proof of a theorem, as described in Section B.1, except that we are allowed to write down any of the hypotheses at any time, just as though they were axioms.

Formally, a proof from hypotheses $H_1, H_2, \ldots, H_k$ is a finite sequence of expressions,

$$S_1, S_2, \ldots, S_n \qquad \text{say,}$$

(which we will call the *steps* of the proof) such that every one of the individual expressions S_i is either an axiom, one of the hypotheses A_i, or follows from earlier members of the sequence by one of the deductive rules.

▶ B.1 Clearly, a proof (in the sense defined in Section B.1) is simply a proof (in the present sense) from no hypotheses, and so what we have just done is widened the meaning of the word. That means that we may write

$$\vdash B$$

to mean that B is a theorem.

Note that the symbol $\vdash$ is *not* part of the formal language we are discussing. It is not even a convenient abbreviation for something that can be said in the language — that is, it is not even part of our semi-formal version. Statements of the form $H_1, H_2, \ldots, H_k \vdash B$ are statements *about* the language, in other words they are part of the *metalanguage* (see below).

We will also write $A \dashv\vdash B$ to mean that both $A \vdash B$ and $B \vdash A$ and say that A and B are *syntactically* (or *deductively*) equivalent.

▶ **2**.C There is more discussion of deductions in Section **2**.C of the next Chapter.

B.5 Metalanguage and the coloured font

In these notes I will make many statements *in* one or other of the various formal languages we discuss, however as more often than not I will make statements *about* them; such statements will contain ideas, words or symbols not expressible in the language itself. We say that such statements are made in the *metalanguage* and, as things get more complicated, the metalanguage tends to get more formal itself. When talking about formal logic, we make statements in (or are doing) *metalogic*, when talking about mathematics we make statements in (or are doing) *metamathematics* and so on.

In the section on Deductions above, all the statements are in the metalanguage. As remarked there, even though $H \vdash B$ for instance looks pretty formal, it is not part of the object language because the symbol $\vdash$ is not a symbol of the language.

For an everyday example, the phrase «Il pleut» is a statement in the French language. In the statements “It is possible to say ‘It is raining’ in French” and “«Il pleut» is French for ‘It is raining’”, we are using English as metaFrench.

For the kind of discussions we will be having, it is essential to distinguish between expressions of the object language and statements made in the metalanguage. We will be dealing with the language of mathematics and languages which express parts of mathematics. These

languages of course contain symbols and expressions which are the same as the ones we use in the metalanguage, providing a rich source of confusion.

For example, suppose we are dealing with a language which contains symbols for the numerals $0, 1, 2, \ldots$, $+$ for addition and $=$ for equality. What are we to make of this?

$$2 + 2 = 4$$

Is it a statement in the object language, saying that 2+2 is equal to 4 (true) or is it a statement in the metalanguage, saying that the expressions "2+2" and "4" are the same *string* (false: the first string has three characters, the second one only one)? The confusion arises of course because there is an ambiguity here about the equality symbol: it could be a symbol of the language itself, or we could be using it as part of our metalanguage.

What we do in everyday English, if we are being careful, is use quotes, something like this:

"2+2=4"	the single expression (true)
"2+2"= "4"	the two strings are the same (false).

Sprinkling mathematics with quote marks does not work very well: it quickly becomes unreadable. Another way to deal with the problem is to use a noticeably different font. In these notes I will use a different font colour, so that expressions in the formal language are easily distinguished — that accounts for the coloured symbols in the text above.

Also, because equality and inequality of expressions *as strings* is often discussed, I will use a special symbol for this, a heavy circled equals sign:

$X \circledequal Y$	X and Y are the same string,
$X \not\circledequal Y$	X and Y are not the same string.

And here is how I would write the "2+2=4" examples:

$2 + 2 = 4$	the single expression (true)
$2 + 2 \circledequal 4$	the two strings are the same (false).

When we look at important formal languages in detail, we will find that they are enormously long-winded. So we introduce many abbreviations to make them easier to read and deal with. We have already done so with our simple $\mathbf{I} - \mathbf{a} - \mathbf{b}$ language with the abbreviation $\mathbf{a}^3$ for $\mathbf{aaa}$. Such abbreviations will also be written in the "formal language font colour". In discussing that simple system, we also used expressions such as $x\mathbf{a}^3y$. Here the x and y are not actually symbols of the language; they stand for strings in the language. We write them in the formal font colour because they represent something that is entirely in the formal language. Discussions above of strings X and Y follow the same pattern.

So here is the general rule followed in these notes:

- Anything that is entirely in the formal font colour is, or represents, a string or strings in the formal language itself.

- Anything that is entirely in normal black is in the metalanguage (or just generally rabbiting on).
- Anything that is a mixture of the two colours is in the metalanguage, discussing the indicated bits of the formal language.

Note that the symbols $\ominus$ and $\not\ominus$ are never part of the formal language. They belong to our metalanguage. The same is true of the symbols $\vdash$ and $\nvdash$ discussed earlier.

So, excuse me if I go on a bit about this coloured font business. The point of using it is to make clear, as far as possible, what is an expression in the formal language under discussion and what is not. For example, when writing about the toy formal language of Section A.2, I wrote

► A.2

- **laaa** in colour, because it is a genuine expression of the language;
- $(\mathbf{ab}^2)^3$ in colour, because it is an abbreviation for a genuine expression of the language, namely **abbabbabb**;
- **xaaay** in colour, because the letters **x** and **y** are stated to represent expressions, and so the whole thing does.

On the other hand, supposing that we give the toy language a name, $\mathcal{T}$ say, that would not be coloured. It is not an expression, nor does it represent one: it represents the language itself.

I have already pointed out that, in statements such as

$$x > 0 \quad \vdash \quad x + y > y$$

there are two formal expressions connected by a metasymbol. The entire statement is not a formal expression. Where colour coding is used, the whole statement must be coloured to be a formal expression.

I will try to use this colour convention as consistently as possible throughout these notes, except for Chapters 5 and 6. Virtually everything that looks formal in those Chapters is formal (or semiformal), so the colouring would just be a nuisance there.

2. SENTENTIAL LOGIC

A A language for Sentential Logic

A.1 The symbols

We will set up a language, which I will call **SL**, which contains Sentential Logic in a particularly simple form: a language which is designed to express Sentential Logic, and nothing else. We start with the symbols. These are of three kinds:

The *connective symbols*	$\neg$	$\Rightarrow$		
The *proposition symbols*	p	q	r	and so on
The *punctuation symbols*	$($	$)$		

A.2 Comments

(1) What happened to the other connectives, $\wedge$, $\vee$ and $\Leftrightarrow$ for instance? They have disappeared in our program of simplifying the language. All the connectives of Sentential Logic can be expressed in terms of the two above, for example $p \vee q$ is the same as $(\neg p) \Rightarrow q$. Thus the other connectives are redundant: anything that can be expressed in Sentential Logic can be expressed using only the two symbols above.

(2) What do we mean by "and so on" for the proposition symbols? Well, first we could assume that there is only a finite number of letters, say four which is enough for most purposes, or even 26 if you like. But that will not be satisfactory, because it turns out that in some of our manipulations we will want to be able to add extra propositional symbols whenever necessary, so we assume that there is always an infinite number of propositional symbols, usually denumerably infinite. Does this make sense? Can we actually have an infinite number of different symbols, each of which is recognisably different from every other one? Yes we can, so long as we interpret "infinite" as meaning "potentially infinite" in the sense that we can go on writing distinct symbols down as long as we like without ever reaching a dead end. Here is an example:

V W VV VVV and so on.

Alternatively, we could simply use two symbols, for instance p and $'$, which would allow us our unending set of proposition symbols

$$p \quad p' \quad p'' \quad p''' \quad \ldots$$

In a sense the last suggestion is the simplest of all, so why don't we make this part of the original definition, in keeping with our program? We want the language to be simple in the sense that it will be easy to define and easy to prove things about it. Saying that we have

an infinite number of proposition symbols if we like makes the language easier to read and write and does not in fact put any extra difficulties in our way.

So we see that there are in fact many versions of the language **SL**, depending upon how many proposition symbols and which particular ones we choose to use. Mostly the difference between these versions of the language is trivial, we ignore it and speak of *the* language **SL**.

A.3 Expressions

(S1) Any proposition symbol (by itself) constitutes a expression.

(S2) If A is a expression, then so is $(\neg A)$.

(S3) If A and B are expressions, then so is $(A \Rightarrow B)$.

A.4 Comments

(1) This definition should perhaps be put slightly more carefully. It is meant to state that expressions can be built up from simpler ones in this way, and only in this way. One can think of this as a set of construction rules, telling one how it is allowed to build up expressions. This way of looking at it does not make it obvious that we have a decision algorithm here. Perhaps a better approach is to notice that, in the second and third rules, the new expression is longer than the one(s) from which it is formed. We can therefore reword the definition in a recursive fashion: a string C is a expression if and only if it is of one of the forms

(S1′) It is a lone proposition symbol.

(S2′) There is a *shorter* expression A such that $C \ominus (\neg A)$.

(S3′) There are *shorter* expressions A and B such that $C \ominus (A \Rightarrow B)$.

(2) The rules introduce more parentheses than we usually use. However if the parentheses in (S3) were omitted, for instance, we would be allowed to write $p \Rightarrow q \Rightarrow r$, which is meaningless (since implication is not associative). Our normal usage depends on complicated rules of precedence in the application of operators, which presumably we could include in our rules, at the expense of making our definition much more complicated. The easier approach is to insist on a complete set of parentheses every time, at the expense of legibility — for example, the expression we would normally write $(p \Rightarrow \neg p) \Rightarrow \neg p$ must now be written $((p \Rightarrow (\neg p)) \Rightarrow (\neg p))$.

(3) In our semi-formal version, we will of course allow ourselves to dispense with parentheses according to the usual conventions. We will also allow ourselves to use the other connectives as abbreviations for the appropriate constructions in the formal language. Therefore, when in the semi-formal language we write $p \vee q$ we are only using a convenient shorthand for the true formal $((\neg p) \Rightarrow q)$.

You should convince yourself you can construct formal equivalents for the semi-formal $p \wedge q$ and $p \Leftrightarrow q$.

That completes the definition of the language. Of course we have a theory in mind, namely the set of all *tautologies*, the logical formulas which are always true, irrespective of the truth or falsity of the propositions the individual proposition symbols represent. Well-known examples are

$$p \wedge q \Leftrightarrow q \wedge p$$
$$(p \Rightarrow (q \wedge \neg q)) \Rightarrow \neg p$$
$$\neg(p \wedge q) \Leftrightarrow (\neg p) \vee (\neg q)\,.$$

[These are, of course, presented in semi-formal version. Try translating them into formal language to see why we don't want to use it much.]

A.5 Induction over construction of an expression

Continuing the idea of Comment (1) above, it is possible to prove things about expressions by induction over their length. To prove something is true of all expressions of **SL**, it is sufficient to prove:

(1) It is true of all expressions which consist of a single proposition symbol;

(2) for any expression A, if it is true of A then it is also true of $(\neg A)$; and

(3) for any expressions A and B, if it is true of both of them then it is also true of $(A \Rightarrow B)$.

This process is called *induction over the construction* of the expression. We will be using it from time to time.

A.6 The axioms

We will define the theory deductively. We start by defining the axioms.

We select three *axiom schemata*:

(A1) $(A \Rightarrow (B \Rightarrow A))$

(A2) $((A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C)))$

(A3) $(((\neg A) \Rightarrow (\neg B)) \Rightarrow (((\neg A) \Rightarrow B) \Rightarrow A))$

As before, to say that these are axiom schemata means that each represents an infinite number of actual axioms, created by substituting any actual expressions for the symbols A, B and C used here. For example, as instances of (A1) are the actual axioms

$$(p \Rightarrow (q \Rightarrow p))$$
$$(p \Rightarrow (p \Rightarrow p))$$
$$(((\neg q) \Rightarrow (\neg p)) \Rightarrow ((p \Rightarrow (\neg p)) \Rightarrow ((\neg q) \Rightarrow (\neg p))))$$

and of course infinitely many others. It is a straightforward, if tiresome, process to decide whether any given expression is an axiom, that is, an instance of one of the three schemata

above. Perhaps a simpler way of looking at this is to read (A1) as saying that any expression *of the form* $(A \Rightarrow (B \Rightarrow A))$ is an axiom, and to read (A2) and (A3) in a similar fashion.

The axioms are a little easier to comprehend if we remove some parentheses and write them semi-formally:

(A1′) $A \;\Rightarrow\; (B \Rightarrow A)$

(A2′) $\big(A \Rightarrow (B \Rightarrow C)\big) \;\Rightarrow\; \big((A \Rightarrow B) \Rightarrow (A \Rightarrow C)\big)$

(A3′) $(\neg A \Rightarrow \neg B) \;\Rightarrow\; \big((\neg A \Rightarrow B) \Rightarrow A\big)$

(There's a few other visual clues here too — extra space and some larger parentheses.)

A.7 The rules

Finally, we need our deductive rules. In fact, we need only one, called *Modus Ponens*. It is also a schema:

$$(MP) \qquad \frac{A\ ,\ (A \Rightarrow B)}{B}$$

that is, given A and $(A \Rightarrow B)$ one can deduce B. As remarked in the previous chapter, what this really means is that an expression is a member of the theory (i.e. a theorem) if and only if there is a finite sequence of statements, $S_1, S_2, \ldots, S_n$ say, such that every one is either an axiom, or can be produced from two earlier ones in the sequence by an application of modus ponens, and in which the last one S_n is in fact B; The sequence $S_1, S_2, \ldots, S_n$, then, is a *formal proof* of B.

A.8 Comments on the axioms and rules

(i) This is not the only way to axiomatise Sentential Logic. A brief look at textbooks on logic, especially symbolic logic or mathematical logic, will yield many other ways of doing it. It may even seem to be a bit strange — after all, the axioms (A2) and (A3) are not enormously immediately compellingly obvious, especially compared with such gems as $(A \wedge B) \Leftrightarrow (B \wedge A)$. It is, however, wonderfully economical: we get the whole of Sentential Logic in three reasonably simple axiom schemata and one very simple rule of deduction. This is a great help later. (We look at some other axiomatisations in Section H.)

▶ H

(ii) It has already been pointed out that we do not (yet) have a symbol for the logical "and" connective; we will introduce it later in Section F.6. When we do, we will also prove that $A \Rightarrow (B \Rightarrow C)$ is equivalent to $A \wedge B \;\Rightarrow\; C$ (in Theorem F.9).

▶ F.6
▶ F.9

However, if we allow ourselves to look forward to this notation, we can rewrite the three axioms thus:

(A1″) $A \wedge B \;\Rightarrow\; A$

(A2″) $(A \wedge B \Rightarrow C) \;\wedge\; (A \Rightarrow B) \;\Rightarrow\; (A \Rightarrow C)$

(A3″) $(\neg A \Rightarrow \neg B) \;\wedge\; (\neg A \Rightarrow B) \;\Rightarrow\; A$

They perhaps make better sense in this form.

(iii) It is by no means obvious that we can in fact deduce the whole of Sentential Logic from this basis. Try, for instance, discovering a proof of the very simple theorem $p \Rightarrow p$ (proof on the next page: no peeking!).

(iv) It is possible to avoid the "schemata" way of setting up axioms. An alternative approach is to define (A1), (A2),(A3) as axioms as follows (here p, q and r are proposition symbols):

(A1 alt) $(p \Rightarrow (q \Rightarrow p))$

(A2 alt) $((p \Rightarrow (q \Rightarrow r)) \Rightarrow ((p \Rightarrow q) \Rightarrow (p \Rightarrow r)))$

(A3 alt) $(((\neg p) \Rightarrow (\neg q)) \Rightarrow (((\neg p) \Rightarrow q) \Rightarrow p))$

at the cost of adding one more *substitution rule*, which states that, substituting any expression for any proposition symbol throughout any theorem (consistently) yields another theorem. (MP must remain a schema, for reasons explained above.)

(v) It is possible to define this theory semantically; we will look at this later. Also the "truth table" method yields a decision process without too much trouble: an algorithm which will decide whether any given expression is a theorem or not. Once we have checked this, we will have proved that the theory is decidable.

B Examples of formal proofs

First, a formal proof of a theorem.

B.1 Theorem

$p \Rightarrow p$

Proof.

$((p \Rightarrow ((p \Rightarrow p) \Rightarrow p)) \Rightarrow ((p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p)))$
$(p \Rightarrow ((p \Rightarrow p) \Rightarrow p))$
$((p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p))$
$(p \Rightarrow (p \Rightarrow p))$
$(p \Rightarrow p)$ ■

That is the full formal proof. It is extremely unfriendly to human readers — on the other hand a computer would have little trouble with it. Below it is presented again in a less formal form. Parentheses have been removed, some spacing has been inserted, line numbers have been added, and a gloss on the right indicating which rules or axioms have been used. In Line 1, the gloss "Ax 2" means that this is an instance of Axiom Schema (A2) and, in line 3, the gloss "MP: 1 and 2" means that this is the result of applying Modus Ponens to Lines 1 and 2.

1	$(p \Rightarrow ((p \Rightarrow p) \Rightarrow p)) \;\Rightarrow\; ((p \Rightarrow (p \Rightarrow p)) \Rightarrow (p \Rightarrow p))$	Ax 2
2	$p \;\Rightarrow\; ((p \Rightarrow p) \Rightarrow p)$	Ax 1
3	$(p \Rightarrow (p \Rightarrow p)) \;\Rightarrow\; (p \Rightarrow p)$	MP: 1 and 2
4	$p \Rightarrow (p \Rightarrow p)$	Ax 1
5	$p \Rightarrow p$	MP: 3 and 4 ■

We could just as easily have proved a schematic version of the theorem:

B.2 Theorem

$A \Rightarrow A$

The proof would be the same as the one above, with A substituted for p throughout.

B.3 Deduction

$p \Rightarrow q\ ,\ q \Rightarrow r \;\vdash\; p \Rightarrow r$

Proof.

1	$(q \Rightarrow r) \;\Rightarrow\; (p \Rightarrow (q \Rightarrow r))$	Ax 1
2	$q \Rightarrow r$	Hyp
3	$p \Rightarrow (q \Rightarrow r)$	MP: 1 and 2
4	$(p \Rightarrow (q \Rightarrow r)) \;\Rightarrow\; ((p \Rightarrow q) \Rightarrow (p \Rightarrow r))$	Ax 2
5	$(p \Rightarrow q) \Rightarrow (p \Rightarrow r)$	MP: 3 and 4
6	$p \Rightarrow q$	Hyp
7	$p \Rightarrow r$	MP: 6 and 5

■

As with Theorem B1, we could just as easily have proved a schematic version:

$$A \Rightarrow B \;,\; B \Rightarrow C \quad \vdash \quad A \Rightarrow C$$

by substituting A, B and C for p, q and r throughout the proof.

C Deductions again

C.1 Using deductions as rules

One important thing about deductions is that, once proved, we can henceforth use them in the same way as deductive rules. For example, having proved the deduction above, we can henceforth proceed as though we had the new deductive rule

$$\frac{A \Rightarrow B\ ,\ B \Rightarrow C}{A \Rightarrow C}\ .$$

What we are doing here is extending our semi-formal language, and making it more convenient. We are not changing the formal language in any way. Using the deduction just proved, for instance, we now allow ourselves to put a line in one of our semi-formal proofs of the form

$$A \Rightarrow C \qquad\qquad \text{By Deduction B2}$$

provided we already have two lines $A \Rightarrow B$ and $B \Rightarrow C$ somewhere. The justification for this new freedom is that there is a recipe for converting such a proof-step into a full formal version for the object language. Here it is:

> Replace the line $A \Rightarrow C$ in the semi-formal version with the entire proof of the deduction B2, as given above. The result is a valid proof. The new lines added to the proof include $A \Rightarrow C$, and so the new proof contains all the lines of the old proof in the same order. Therefore all the old lines are still valid, for the same reasons (except for $A \Rightarrow C$ itself, for which we need a proper reason). We now justify each of the new steps in the proof (and that includes $A \Rightarrow C$). If a new step was an axiom, then it still is, so it is valid. If a new step followed by Modus Ponens from two earlier steps in the proof of the deduction, then these earlier steps have just been added to our new formal proof: the step being examined is valid. Finally, if the new step was a hypothesis of the deduction, then it is a repetition of an earlier step of the old proof, and this is also valid, as remarked at the end of Section **1**.B1.

C.2 The Deduction Theorem

Note Despite its name, the Deduction Theorem is *not* a theorem of **SL**. It is a theorem about this theory and so is a metatheorem. As remarked above, there is nothing in the language **SL** which allows the idea symbolised by $\vdash$ to be expressed, let alone anything to be proved about it (within **SL**).

C.3 The Deduction Theorem (simple form)

$$\text{If} \quad H \vdash B \quad \text{then} \quad \vdash H \Rightarrow B\,.$$

Note It is easy to prove the converse (that if $\vdash H \Rightarrow B$ then $H \vdash B$) — this is almost a restatement of Modus Ponens. Therefore the Deduction Theorem can be restated

$$H \vdash B \quad \text{if and only if} \quad H \Rightarrow B\,.$$

Which means that, in some sense, the symbol $\Rightarrow$ says, *within* the formal language, what the symbol $\vdash$ says *outside* it (in the metalanguage). This corresponds to what we are used to: that one can do formal manipulations with the symbol $\Rightarrow$, yet it can also be interpreted as saying that one thing can be deduced from another. For our present purposes it is important

to keep the two ideas distinct, though the Deduction Theorem states the close relationship between them.

Proof of the Deduction Theorem. We must show that, if there is a formal proof of B from the hypothesis H, then there is a (different) proof of $H \Rightarrow B$ based on no hypotheses.

The recipe. Suppose that we are given a proof of B from H (the *old proof*). We want to construct a proof of $H \Rightarrow B$ (the *new proof*). The old proof consists of a number of lines, each of which is either an axiom, the hypothesis H, or follows from two earlier lines of the old proof by Modus Ponens. We construct the new proof by going through the old one and replacing each step, S say, by several new ones, the last of which is $H \Rightarrow S$.

If the old line was an axiom, we replace it by

1.	S	Ax whichever
2.	$S \Rightarrow (H \Rightarrow S)$	Ax 1
3.	$H \Rightarrow S$	MP: 1 and 2

If the old line was the hypothesis H, we replace it by the entire proof of $H \Rightarrow H$, as given in Theorem B.1 above. ▶ B.1

If the old line S followed by Modus Ponens from two earlier lines in the proof,

$$R \quad \text{and} \quad R \Rightarrow S \quad \text{say,}$$

then in our *new* proof we already have earlier lines

$$H \Rightarrow R \quad \text{and} \quad H \Rightarrow (R \Rightarrow S)\,.$$

We replace S by

1.	$(H \Rightarrow (R \Rightarrow S)) \;\Rightarrow\; ((H \Rightarrow R) \Rightarrow (H \Rightarrow S))$	Ax 2
2.	$(H \Rightarrow R) \Rightarrow (H \Rightarrow S)$	MP: 1 and earlier
3.	$H \Rightarrow S$	MP: 2 and earlier

Proceeding in this way we produce a sequence of statements culminating in $H \Rightarrow B$. By examining the recipe above it is easy to see that it is in fact a proof. ■

C.4 The Deduction Theorem (general form)

If $\quad H_1, H_2, \ldots, H_k \vdash B \quad$ (with $k \geq 1$) $\quad$ then $\quad H_1, H_2, \ldots, H_{k-1} \vdash H_k \Rightarrow B\,.$

Proof. This is a straightforward extension of the proof of the simple case above, and is left as an exercise. ■

Note If we have $H_1, H_2 \vdash B$ then two applications of the Deduction Theorem yields

$$\vdash \; H_1 \Rightarrow (H_2 \Rightarrow B) \; .$$

In the same way, three applications of the theorem to $H_1, H_2, H_3 \vdash B$ eventually yields

$$\vdash \; H_1 \Rightarrow (H_2 \Rightarrow (H_3 \Rightarrow B)) \; .$$

C.5 Example

Earlier in this chapter, we proved that

$$A \Rightarrow B \; , \; B \Rightarrow C \; \vdash \; A \Rightarrow C \, .$$

Using the Deduction Theorem we may now conclude that

$$A \Rightarrow B \; \vdash \; (B \Rightarrow C) \Rightarrow (A \Rightarrow C)$$

and that

$$\vdash \; (A \Rightarrow B) \; \Rightarrow \; ((B \Rightarrow C) \Rightarrow (A \Rightarrow C)) \, .$$

It is a useful exercise to use the recipe given above to construct proofs for the last two statements. It then becomes clear what a great labour-saving device the Deduction Theorem is.

D Other derived rules

There are a number of other tricks for making shorter proofs in the semi-formal language. Remember that each of these owes its livelihood to the existence of a recipe which will convert a proof in which it is used to a proof in which it is not. Therefore it remains an algorithmic, if tedious, process to convert any semi-formal proof into its fully formal counterpart.

D.1 Substitution

This has been described already. If a theorem or deduction has been proved, then any expression may be substituted for its proposition symbols to get another valid theorem or deduction.

The recipe for replacing this trick is to make the same replacement throughout its proof. We avoid the need for substitution by profligate use of schemata, but it is comforting to know it is there.

D.2 Using theorems already proved

If a theorem has been proved, then it may be used as a step in any later proof whenever we like, just as though it were an axiom.

The recipe for producing a proof that does not require this trick is to insert, just before the line in question, its entire proof.

D.3 Using deductions already proved

If a deduction has been proved, it can henceforth be used as a deductive rule in proofs just like Modus Ponens. This has been discussed and justified above.

D.4 Turning theorems into deductions

Any theorem of the form $A \Rightarrow B$ can immediately be used as a deduction $\frac{A}{B}$ in further proofs.

E Subdeductions

We can clean up the proof of $A \Rightarrow B\,,\ B \Rightarrow C \vdash A \Rightarrow C$ even further. Informally, we are tempted to argue thus:

We are given the hypotheses $A \Rightarrow B$ and $B \Rightarrow C$. Now suppose that A (a themporary hypothesis). Then, from the first hypothesis and MP, we have B and then, from that and the second hypothesis, we have C. Since we got this on the supposition A, we have proved that $A \Rightarrow C$.

It will be very useful to make this kind of subsidiary deduction part of our semi-formal system. Let us try to make it look more formal.

Proof 1

1	$A \Rightarrow B$	hyp
2	$B \Rightarrow C$	hyp
3	A	subhyp
4	B	MP: 3 & 1
5	C	MP: 4 & 2
6	$A \Rightarrow C$	Ded: 3–5

Here I have enclosed the subsidiary deduction in an inner box, and for both the main deduction and the subsidiary one have drawn a line to separate the hypotheses from the proper proof steps. It is very important in this kind of proof to know where our subdeductions start and end, and exactly which lines are hypotheses and which are the proper proof steps. These boxes seem to me to be a good way of doing it.

So here we have a subsidiary deduction (Lines 3–5) based on a temporary (or subsidiary) hypothesis A. When that subsidiary proof is finished, we can conclude that $A \Rightarrow C$. The steps in the subsidiary proof are allowed to use any steps that came before, whether in the subsidiary proof or the main one — for example, Line 5 makes use of Lines 4 and 2.

How do we justify this? By showing that a proof which uses such a subsidiary deduction can be replaced by one which does not. The recipe is to make the subsidiary proof into a separate deduction which uses *all* the preceding steps of the main proof as hypotheses. Like this:

Proof 2

1	$A \Rightarrow B$	hyp
2	$B \Rightarrow C$	hyp
3	A	hyp
4	B	MP: 3 & 1
5	C	MP: 4 & 2

Note that the proper steps of the new proof (Lines 4 and 5 in this example) are justified by the same things as they were in Proof 1 above. Proof 2 is, of course, a proof of the deduction

$$A \Rightarrow B\,,\ B \Rightarrow C\,,\ A \vdash C$$

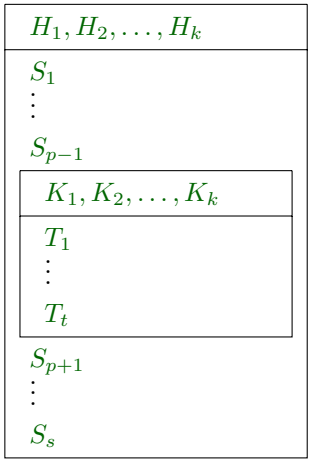
And now the Deduction Theorem gives us

$$A \Rightarrow B\,,\ B \Rightarrow C \ \vdash\ A \Rightarrow C$$

as required.

We need to make sure that this works in general. Suppose then that we have a proof with a subsidiary deduction of this form:

Proof 3



hypotheses for the main proof

subsidiary hypotheses

Here the usual rules apply: the hypotheses $H_1, H_2, \ldots, H_h$ and $K_1, K_2, \ldots, K_k$ can be any expressions at all.

Each of the "main" steps $S_1, S_2, \ldots, S_s$ can be an axiom, one of the main hypotheses $H_1, H_2, \ldots, H_h$ or follow by Modus Ponens from any of its preceding *main* steps S_i. It may *not* use any of $K_1, K_2, \ldots, K_k$ or $T_1, T_2, \ldots, T_t$ for the purposes of Modus Ponens.

Any one of the main steps $S_{p+1} \ldots S_s$ which follow the subsidiary deduction may also be of the form

$$K_1 \ \Rightarrow\ (K_2 \ \Rightarrow\ \ldots \ \Rightarrow\ (K_k \ \Rightarrow\ T_i))\ldots)$$

where T_i is any one of $T_1, T_2, \ldots, T_t$.

Each of the steps $T_1, T_2, \ldots, T_t$ of the subsidiary proof can be an axiom, one of the main hypotheses $H_1, H_2, \ldots, H_h$ or one of the subsidiary ones $K_1, K_2, \ldots, K_k$ or follow by Modus Ponens from any of the *preceding* main steps S_i or subsidiary ones T_i.

Now this purports to be a proof of $H_1, H_2, \ldots, H_h \vdash S_s$; we must show that this is so, by showing that it can be replaced by a proof which does not involve the subsidiary deduction.

Following the idea already used above, we set down a new deduction which is the same as our subsidiary one, except that all of $H_1, H_2, \ldots, H_h$, $S_1, S_2, \ldots, S_{p-1}$ and $K_1, K_2, \ldots, K_k$ are used as hypotheses:—

$H_1, H_2, \ldots, H_k$	hypotheses	
$S_1, S_2, \ldots, S_{p-1}$	more hypotheses	
$K_1, K_2, \ldots, K_k$	more hypotheses	**Proof 4**
T_1		
$\vdots$		
T_t		

Once again, note that anything which justified one of the lines $T_1, T_2, \ldots, T_t$ in Proof 3 still justifies that line in Proof 4, so this is then a valid proof of the deduction

$$H_1, H_2, \ldots, H_h, S_1, S_2, \ldots, S_{p-1}, K_1, K_2, \ldots, K_k \vdash T_i$$

(for any $i = 1, 2, \ldots, t$).

Then by repeated applications of the Deduction Theorem, we have

$$H_1, H_2, \ldots, H_h, S_1, S_2, \ldots, S_{p-1} \vdash K_1 \Rightarrow (K_2 \Rightarrow \ldots \Rightarrow (K_k \Rightarrow T_i)) \ldots)$$

so it is now valid to write down

$$K_1 \Rightarrow (K_2 \Rightarrow \ldots \Rightarrow (K_k \Rightarrow T_i)) \ldots)$$

for any of $S_{p+1}, S_{p+2}, \ldots, S_s$, as required.

We will see how useful this method is in the next few sections. For now, let us just note that it can be made even more powerful in a couple of (obvious?) ways:

- A main proof may have any number of subdeductions and
- subdeductions may be nested inside one another (to any depth).

F Some theorems and deductions

In this section I give a number of theorems and deductions and their proofs. This is by no means an exhaustive list of the theorems of **SL**, not even a list of all the more useful ones. There are enough theorems here, however, to supply a working basis: from this further theorems can be proved without difficulty. The proofs illustrate the techniques discussed above.

F.1 Theorem

$\vdash A \Rightarrow A$

Proof. This has already been provided. ■

F.2 Theorem

(a) $A \Rightarrow B\ ,\ B \Rightarrow C \vdash A \Rightarrow C$

(b) $A \Rightarrow (B \Rightarrow C)\ ,\ B \vdash A \Rightarrow C$

(c) $A\ ,\ \neg A \vdash B$

(d) $A \vdash B \Rightarrow A$

Proof of (a). This has been done to death above in Section E.

Proof of (b)

1	$A \Rightarrow (B \Rightarrow C)$	hyp
2	B	hyp
3	A	subhyp
4	$B \Rightarrow C$	MP: 3 and 1
5	C	MP: 2 and 4
6	$A \Rightarrow C$	Ded: 3–5

Proof of (d) This is simply Axiom A1 converted into a deduction.

Proof of (c)

1	A	hyp
2	$\neg A$	hyp
3	$\neg B \Rightarrow A$	By Part (d)
4	$\neg B \Rightarrow \neg A$	By Part (d)
5	$(\neg B \Rightarrow \neg A) \ \Rightarrow\ ((\neg B \Rightarrow A) \Rightarrow B)$	Ax 3
6	$(\neg B \Rightarrow A) \Rightarrow B$	MP: 4 & 5
7	B	MP: 3 & 6

■

F.3 Theorem

$$A \vdash\!\dashv \neg\neg A$$

Note that this theorem has a number of immediate corollaries:

$$A \vdash \neg\neg A$$
$$\neg\neg A \vdash A$$
$$\vdash A \Rightarrow \neg\neg A$$
$$\vdash \neg\neg A \Rightarrow A$$

and later, when we have dealt with equivalence,

$$\vdash A \Leftrightarrow \neg\neg A$$

Many of the subsequent theorems have the same sort of corollaries; to save space they will not all be listed.

Proof of Theorem F.3. This is an equivalence, requiring two separate proofs. We show first that $\neg\neg A \vdash A$.

1	$\neg\neg A$	hyp
2	$\neg A \Rightarrow \neg\neg A$	Theorem F.2(d) on 1
3	$(\neg A \Rightarrow \neg\neg A) \;\Rightarrow\; ((\neg A \Rightarrow \neg A) \Rightarrow A)$	Ax 3
4	$(\neg A \Rightarrow \neg A) \Rightarrow A$	MP: 2 and 3)
5	$\neg A \Rightarrow \neg A$	Theorem F.1
6	A	MP: 4 and 5

Next we show that $A \vdash \neg\neg A$.

1	A	hyp
2	$\neg\neg\neg A \Rightarrow \neg A$	Part 1 of this proof
3	$(\neg\neg\neg A \Rightarrow \neg A) \;\Rightarrow\; ((\neg\neg\neg A \Rightarrow A) \Rightarrow \neg\neg A)$	Ax 3
4	$(\neg\neg\neg A \Rightarrow A) \Rightarrow \neg\neg A$	MP: 2 and 3)
5	$\neg\neg\neg A \Rightarrow A$	Theorem F.2(d) on 1
6	$\neg\neg A$	MP: 5 and 4

■

F.4 Theorem

(a) $A \Rightarrow B \;\vdash\!\dashv\; \neg B \Rightarrow \neg A$

(b) $A \Rightarrow \neg B \;\vdash\!\dashv\; B \Rightarrow \neg A$

(c) $\neg A \Rightarrow B \;\vdash\!\dashv\; \neg B \Rightarrow A$

Proof of (a). First we show that $\neg B \Rightarrow \neg A \vdash A \Rightarrow B$.

1	$\neg B \Rightarrow \neg A$	hyp
2	$(\neg B \Rightarrow \neg A) \;\Rightarrow\; ((\neg B \Rightarrow A) \Rightarrow B)$	Ax 3
3	$(\neg B \Rightarrow A) \Rightarrow B$	MP: 1 and 2
4	A	subhyp
5	$\neg B \Rightarrow A$	Theorem F.2(d) on 4
6	B	MP:5 and 3
7	$A \Rightarrow B$	Ded: 4–6

Next we show that $A \Rightarrow B \vdash \neg B \Rightarrow \neg A$.

1	$A \Rightarrow B$	hyp
2	$\neg\neg A \Rightarrow A$	Theorem F.3
3	$\neg\neg A \Rightarrow B$	Theorem F.2(a) on 1 and 2
4	$B \Rightarrow \neg\neg B$	Theorem F.3 again
5	$\neg\neg A \Rightarrow \neg\neg B$	F.2(a) on 3 and 4
6	$(\neg\neg A \Rightarrow \neg\neg B) \;\Rightarrow\; (\neg B \Rightarrow \neg A)$	First part of this proof
7	$\neg B \Rightarrow \neg A$	MP: 5 and 6

Proofs of (b) and (c) as exercises. ■

F.5 Theorem

(a) $A \Rightarrow B \;,\; \neg B \;\vdash\; \neg A$

(b) $\neg(A \Rightarrow B) \;\vdash\; A$

(c) $\neg(A \Rightarrow B) \;\vdash\; \neg B$

Proofs as easy exercises.

F.6 Notation

Now let us introduce the other connectives. As already stated, they are not part of the formal language, but are used in the semi-formal one as abbreviations for the more complicated expressions which they represent.

$A \wedge B$	stands for	$\neg(A \Rightarrow \neg B)$
$A \vee B$	stands for	$\neg A \Rightarrow B$
$A \Leftrightarrow B$	stands for	$(A \Rightarrow B) \wedge (B \Rightarrow A)$.

F.7 Theorem

(a) $A \wedge B \Rightarrow A$

(b) $A \wedge B \Rightarrow B$

(c) $A\ ,\ B \vdash A \wedge B$

Proof of (a). We have to show that $\neg(A \Rightarrow \neg B) \vdash A$. Note that Theorem F.2(c) can be rewritten $\neg A\ ,\ A \vdash B$ which, by the Deduction Theorem, gives $\vdash \neg A \Rightarrow (A \Rightarrow B)$. Substituting $\neg B$ for B in this starts the proof:

1	$\neg(A \Rightarrow \neg B)$	hyp
2	$\neg A \Rightarrow (A \Rightarrow \neg B)$	As above
3	$\neg(A \Rightarrow \neg B) \ \Rightarrow\ A$	Theorem F.4(c) as deduction on 2
4	A	MP: 1 and 3

Proof of (b) (Note that this is not obvious because we have not yet proved that $\wedge$ is commutative.) We want to prove that $\neg(A \Rightarrow \neg B) \vdash B$.

1	$\neg(A \Rightarrow \neg B)$	hyp
2	$\neg B \Rightarrow (A \Rightarrow \neg B)$	Ax 1
3	$\neg(A \Rightarrow \neg B) \ \Rightarrow\ B$	Theorem F.4(c) as deduction on 2
4	B	MP: 1 and 3

Proof of (c) as an exercise. ■

F.8 Theorem

(a) $A \Rightarrow A \vee B$

(b) $B \Rightarrow A \vee B$

Proof of (a). This is $A \vdash \neg A \Rightarrow B$ and so, by the Deduction Theorem, it is enough to prove that $A\ ,\ \neg A \vdash B$; this is Theorem F.2(c).

Proof of (b) This is Axiom A1 in disguise. ■

F.9 Theorem

(a) $A \Rightarrow (B \wedge C) \ \vdash\ A \Rightarrow B$

(b) $A \Rightarrow (B \wedge C) \ \vdash\ A \Rightarrow C$

(c) $A \Rightarrow B\ ,\ A \Rightarrow C \ \vdash\ A \Rightarrow (B \wedge C)$

(d) $A \Rightarrow (B \Rightarrow C) \ \dashv\vdash\ A \wedge B \ \Rightarrow\ C$

Proof. (a) We prove that $A \Rightarrow (B \wedge C)\ ,\ A \vdash B$.

1.	$B \wedge C$	MP from both hyps
2.	B	Theorem F.7(a) on 1 ■

(b) and **(c)** follow in the same way from Theorems F.5(b) and (c). ▶ F.5

(d) This is an equivalence so as usual there are two things to prove. First we prove $A \Rightarrow (B \Rightarrow C) \vdash A \wedge B \Rightarrow C$.

1	$A \Rightarrow (B \Rightarrow C)$	hyp
2	$A \wedge B$	subhyp
3	A	Theorem F.7(a) on 2
4	$B \Rightarrow C$	MP: 3 and 1
5	B	Theorem F.7(b) on 2
6	C	MP: 5 and 4
7	$A \wedge B \Rightarrow C$	Ded: 2–6

Now we prove $A \wedge B \Rightarrow C \vdash A \Rightarrow (B \Rightarrow C)$. For the first time we will use a doubly-nested subdeduction.

1	$A \wedge B \Rightarrow C$	hyp
2	A	subhyp
3	B	subsubhyp
4	$A \wedge B$	Theorem F.7(c) on 2 and 3
5	C	MP: 4 and 1
6	$B \Rightarrow C$	Ded: 3–5
7	$A \Rightarrow (B \Rightarrow C)$	Ded: 2–6

F.10 Theorem

(a) $(A \vee B) \Rightarrow C \vdash A \Rightarrow C$

(b) $(A \vee B) \Rightarrow C \vdash B \Rightarrow C$

(c) $A \Rightarrow C\ ,\ B \Rightarrow C \vdash (A \vee B) \Rightarrow C$

(d) $A \vee B\ ,\ A \Rightarrow C\ ,\ B \Rightarrow C \vdash C$

(e) $A \vee B\ ,\ \neg A \vdash B$

(f) $A \vee B\ ,\ \neg B \vdash A$

Proof. (a) We want to show that $(\neg A \Rightarrow B) \Rightarrow C \vdash A \Rightarrow C$.

1	$(\neg A \Rightarrow B) \Rightarrow C$	hyp
2	A	subhyp
3	$A \Rightarrow (\neg A \Rightarrow B)$	Apply the Deduction Theorem to Theorem F.2(c)
4	$\neg A \Rightarrow B$	MP: 2 and 3
5	C	MP: 4 and 1

(b) We show that $(\neg A \Rightarrow B) \Rightarrow C \vdash B \Rightarrow C$.

1	$(\neg A \Rightarrow B) \Rightarrow C$	hyp
2	B	subhyp
3	$\neg A \Rightarrow B$	Theorem F.2(d) on 2
4	C	MP: 3 and 1
5	$B \Rightarrow C$	Ded: 2–4

(c) We show that $A \Rightarrow C$, $B \Rightarrow C \vdash (\neg A \Rightarrow B) \Rightarrow C$.

1	$A \Rightarrow C$	hyp
2	$B \Rightarrow C$	hyp
3	$\neg A \Rightarrow B$	subhyp
4	$\neg A \Rightarrow C$	Theorem F.2(a) on 2 and 4
5	$\neg C \Rightarrow A$	Theorem F.4(c) on 4
6	$\neg C \Rightarrow \neg A$	Theorem F.4(a) on 1
7	C	Ax 3 as deduction on 5 and 6
8	$(\neg A \Rightarrow B) \Rightarrow C$	Ded: 3–7

(d) is an immediate corollary of **(c)**. Theorems **(e)** and **(f)** follow immediately from the fact that $A \vee B$ is defined to mean $\neg A \Rightarrow B$. ■

F.11 Comment

The last two groups of deductions encapsulate the functions of $\wedge$ and $\vee$: having proved these, all properties of these connectives follow from these six deductions and no further reference to the rather strange definitions is required. The symmetry between $\wedge$ and $\vee$ will be observed.

▶ F.10 Theorem F.10(d) is called the "Constructive Dilemma". It is a very common way of arguing, encapsulating an argument by cases. Here A and B represent the two cases, and the theorem says that if one or other of the two cases is true and C can be proved in either case, then C is true. It is easily extended to cope with three or more cases.

▶ F.7 Note that Deductions F.7(a) to (c) can be applied to the definition of $\Leftrightarrow$ to yield ...

F.12 Theorem

(a) $(A \Leftrightarrow B) \vdash (A \Rightarrow B)$

(b) $(A \Leftrightarrow B) \vdash (B \Rightarrow A)$

(c) $(A \Rightarrow B)\ ,\ (B \Rightarrow A) \vdash (A \Leftrightarrow B)$

Theorems F.1, F.12(c) and F.2 can be regarded as saying that the relation of implication is a partial order — well . . . a preorder for which the equivalence relation is the equivalence in our language — and, of course, we have to agree to interpret our implication and equivalence symbols as relations on expressions to say this. So it would be best to take this remark as fairly imprecise. In the same way, the following bunch of theorems could be read as saying that our equivalence symbol behaves the way an equivalence relation should.

▶ F.1
▶ F.12
▶ F.2

F.13 Theorem

(a) $\vdash A \Leftrightarrow A$

(b) $A \Leftrightarrow B \vdash B \Leftrightarrow A$

(c) $A \Leftrightarrow B\ ,\ B \Leftrightarrow C \vdash A \Leftrightarrow C$

From here on many of the proofs are quite easy and will not be given.

F.14 Theorem

(a) $A \Leftrightarrow A' \vdash \neg A \Leftrightarrow \neg A'$

(b) $A \Leftrightarrow A'\ ,\ B \Leftrightarrow B' \vdash (A \Rightarrow B) \Leftrightarrow (A' \Rightarrow B')$

(c) $A \Leftrightarrow A'\ ,\ B \Leftrightarrow B' \vdash (A \wedge B) \Leftrightarrow (A' \wedge B')$

(d) $A \Leftrightarrow A'\ ,\ B \Leftrightarrow B' \vdash (A \vee B) \Leftrightarrow (A' \vee B')$

(e) $A \Leftrightarrow A'\ ,\ B \Leftrightarrow B' \vdash (A \Leftrightarrow B) \Leftrightarrow (A' \Leftrightarrow B')$

By repeated applications of these theorems we can substitute equivalent expressions for one another anywhere to yield larger equivalent expressions. This is expressed by the

F.15 Substitution of Equivalents theorem — actually a metatheorem

If A, A' and B are expressions such that $A \Leftrightarrow A'$ and A occurs as a part of B, and we write B' for the result of substituting A' for A in B, then $B \Leftrightarrow B'$.

[Note: it is possible that there may be several occurrences of A in B. In this case the theorem does *not* require that the substitution should be carried out consistently on all occurrences. In may be carried out on some and others left as they were, and the conclusion still holds.]

A similar result for implication holds for two connectives:

F.16 Theorem

(a) $A \Rightarrow A'\ ,\ B \Rightarrow B' \ \vdash\ (A \wedge B) \Rightarrow (A' \wedge B')$

(b) $A \Rightarrow A'\ ,\ B \Rightarrow B' \ \vdash\ (A \vee B) \Rightarrow (A' \vee B')$

It is now easy to prove some basic algebraic-style properties of conjunction and disjunction. Firstly, they are commutative,

F.17 Theorem

(a) $A \wedge B \ \dashv\vdash\ B \wedge A$

(b) $A \vee B \ \dashv\vdash\ B \vee A$

and associative,

F.18 Theorem

(a) $(A \wedge B) \wedge C \ \dashv\vdash\ A \wedge (B \wedge C)$

(b) $(A \vee B) \vee C \ \dashv\vdash\ A \vee (B \vee C)$

Because of these laws we can perform the usual operations on expressions involving repeated conjunctions and disjunctions. We also adopt the usual abbreviations in our semi-formal language:

$A \wedge B \wedge C$ stands for $(A \wedge B) \wedge C$
$A \wedge B \wedge C \wedge D$ stands for $((A \wedge B) \wedge C) \wedge D$ and so on,
and the same for disjunction.

Conjunction and disjunction distribute over each other:

F.19 Theorem

(a) $A \wedge (B \vee C) \ \dashv\vdash\ (A \wedge B) \vee (A \wedge C)$ and $(B \vee C) \wedge A \ \dashv\vdash\ (B \wedge A) \vee (C \wedge A)$

(b) $A \vee (B \wedge C) \ \dashv\vdash\ (A \vee B) \wedge (A \vee C)$ and $(B \wedge C) \vee A \ \dashv\vdash\ (B \vee A) \wedge (C \vee A)$

Some laws which are not like ordinary algebraic laws are *De Morgan's Laws*,

F.20 Theorem

(a) $\neg(A \wedge B) \ \dashv\vdash\ \neg A \vee \neg B$

(b) $\neg(A \vee B) \ \dashv\vdash\ \neg A \wedge \neg B$

and the *Absorptive Laws*,

F.21 Theorem

(a) $A \vee (A \wedge B) \vdash A$

(b) $A \wedge (A \vee B) \vdash A$

Finally, there is the *Law of the Excluded Middle* and the *Law of Dichotomy*:

F.22 Theorem

(a) $\vdash \neg(A \wedge \neg A)$

(b) $\vdash A \vee \neg A$

We will need the very simple facts:

F.23 Theorem

(a) $A \wedge A \vdash A$

(b) $A \vee A \vdash A$

Note that De Morgan's laws give us slight, but useful, variations on a number of the foregoing theorems. Two such we will need are

F.24 Theorem

(a) $\neg A \vdash \neg(A \wedge B)$ and $\neg B \vdash \neg(A \wedge B)$

(b) $\neg A\ ,\ \neg B \vdash \neg(A \vee B)$

F.25 Theorem

(a) $A \Leftrightarrow B \vdash \neg A \Leftrightarrow \neg B$

(b) $A \Leftrightarrow \neg B \vdash \neg A \Leftrightarrow B$

F.26 Fun with T and F

For the purposes of this discussion it will be convenient to define an *antitheorem* to be an expression that is proveably false; that is, B is an antitheorem if $\vdash \neg B$. For example, $B \wedge \neg(A \Rightarrow B)$ is an antitheorem.

Now, choose your favourite theorem, preferably a very simple one (for instance, I would choose $p \Rightarrow p$, where p is some fixed chosen letter of the alphabet); call it **T** for short.

In the same way, choose a nice simple antitheorem ($\neg(p \Rightarrow p)$ is convenient), and call it **F**.

Now, if A is any theorem, we have both $\vdash A$ and $\vdash \mathbf{T}$. From this it is easy to prove that $\vdash A \Leftrightarrow \mathbf{T}$. Using Substitution of Equivalents (F.15), we now know that, wherever any theorem occurs as a subexpression of some larger exprssion, we may replace it by **T**. ▶ F.15

For example, we know that $\vdash (A \wedge B) \Rightarrow A$ and so we can replace it by T in, say,

$$(A \vee B) \;\Rightarrow\; ((A \wedge B) \Rightarrow A) \wedge C)$$

to get the *equivalent* expression

$$(A \wedge B) \;\Rightarrow\; (\mathsf{T} \wedge C)\,. \tag{–1}$$

In the same way, we may substitute F for any antitheorem, wherever it occurs as a subexpression. For example, we can prove that $\vdash \neg(A \wedge \neg(B \Rightarrow A))$, which is to say that $(A \wedge \neg(B \Rightarrow A))$ is an antitheorem. So, making the substitution in

$$(A \Rightarrow (A \wedge \neg(B \Rightarrow A))) \;\Rightarrow\; \neg A$$

we find that it is equivalent to

$$(A \Rightarrow \mathsf{F}) \;\Rightarrow\; \neg A\,. \tag{–2}$$

A good deal of the manipulation we do of expressions in Sentential Logic is concerned with simplifying expressions — replacing expressions by simpler but equivalent ones. In this process it is often useful to replace theorems by T and antitheorems by F, as described above, and then use the various simplifications listed in the next theorem.

For example, using the next theorem we see that the expression (–1) above simplifies to

$$(A \wedge B) \;\Rightarrow\; C$$

and expression (–2) to

$$\neg A \Rightarrow \neg A\,,$$

which we immediately recognise as theorem.

F.27 Theorem

(a) $\neg\mathsf{T} \;\Leftrightarrow\; \mathsf{F}$ and $\neg\mathsf{F} \;\Leftrightarrow\; \mathsf{T}$

(b) $(\mathsf{T} \Rightarrow A) \;\Leftrightarrow\; A$ and $(\mathsf{F} \Rightarrow A) \;\Leftrightarrow\; \mathsf{T}$

(c) $(A \Rightarrow \mathsf{T}) \;\Leftrightarrow\; \mathsf{T}$ and $(A \Rightarrow \mathsf{F}) \;\Leftrightarrow\; \neg A$

(d) $A \wedge \mathsf{T} \;\Leftrightarrow\; A$ and $A \wedge \mathsf{F} \;\Leftrightarrow\; \mathsf{F}$

(e) $A \vee \mathsf{T} \;\Leftrightarrow\; \mathsf{T}$ and $A \vee \mathsf{F} \;\Leftrightarrow\; A$

(f) $(A \Leftrightarrow \mathsf{T}) \;\Leftrightarrow\; A$ and $(A \Leftrightarrow \mathsf{F}) \;\Leftrightarrow\; \neg A$

G Decidability of Sentential Logic

In this section we prove that **SL** is decidable.

G.1 Preliminaries

It was pointed out at the beginning of this chapter that there are in fact many versions of **SL**, depending upon how many symbols and which particular ones we choose to use. For the present purposes we need to know that any expression A that can be proved in any version of **SL** can be proved in a (possibly smaller) version that uses only the proposition symbols that actually occur in A .

G.2 Theorem

Let A be an expression of (a version of) **SL** and suppose that $p_1, p_2, \ldots, p_k$ are proposition symbols of the language which amongst them contain all the proposition symbols that occur in A (here $p_1, p_2, \ldots, p_k$ may or may not be all of the proposition symbols of the language). Then $\vdash A$ in the given version of **SL** if and only if $\vdash A$ in the smaller version whose only proposition symbols are $p_1, p_2, \ldots, p_k$.

Proof. We note first that $k \geq 1$, because every expression contains at least one proposition symbol. Let us temporarily write $\mathbf{SL}_{\text{big}}$ and $\mathbf{SL}_{\text{small}}$ for the two languages. It is obvious that $\mathbf{SL}_{\text{small}}$ consists of exactly those strings and expressions of $\mathbf{SL}_{\text{big}}$ which contain no proposition symbols other than $p_1, p_2, \ldots, p_k$. If $\vdash A$ in $\mathbf{SL}_{\text{small}}$, then there is a proof of A in that language, and that is also a proof in $\mathbf{SL}_{\text{big}}$, so then $\vdash A$ in $\mathbf{SL}_{\text{big}}$ also. Now suppose that $\vdash A$ in $\mathbf{SL}_{\text{big}}$. Then there is a proof, $C_1, C_2, \ldots, C_n$ say, of A in $\mathbf{SL}_{\text{big}}$. This might not be a proof in $\mathbf{SL}_{\text{small}}$, because it may mention proposition symbols other than the allowed ones. However we can manufacture a proof by changing each line C_i of the given proof by replacing every disallowed proposition symbol by p_1. It is easy to see that this operation converts a proof into a proof, that the new proof is a proof in $\mathbf{SL}_{\text{small}}$ and that it has no effect on the last line, which is A. ■

This theorem tells us that, in order to decide whether a expression A in SL (with any number of proposition symbols, finite or infinite) is a theorem or not, it is enough to decide it in the language which consists of only the (finitely many) symbols which actually occur in A — or, if we wish, in any language with those symbols plus a few others. Therefore, for the remainder of this section, we assume that we are working in a language which contains only a finite number of proposition symbols $p_1, p_2, \ldots, p_k$.

G.3 Truth tables

First I describe the algorithm for deciding whether an expression of **SL** is a theorem or not. The description will be informal, and the proof that it works as claimed will be given subsequently, occupying the remainder of this section.

Consider first the connective $\wedge$. The motivating idea of this connective is that the expression $p \wedge q$ should be true when both p and q are true, and false otherwise. We can encapsulate this in a table.

p	q	$p \wedge q$
1	1	1
1	0	0
0	1	0
0	0	0

Here I have used 1 and 0 to denote TRUE and FALSE. Here are the tables for the five basic connectives:

p	$\neg p$
1	0
0	1

p	q	$p \wedge q$
1	1	1
1	0	0
0	1	0
0	0	0

p	q	$p \vee q$
1	1	1
1	0	1
0	1	1
0	0	0

p	q	$p \Rightarrow q$
1	1	1
1	0	0
0	1	1
0	0	1

p	q	$p \Leftrightarrow q$
1	1	1
1	0	0
0	1	0
0	0	1

One can construct a table for more complicated expressions by a mechanical process. For example, to construct a truth table for the expression $(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$, start with an empty table, in which there is a column corresponding to each of its sub-expressions, building up from left to right:

p	q	$p \Rightarrow q$	$\neg p$	$\neg q$	$\neg q \Rightarrow \neg p$	$(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$
1	1					
1	0					
0	1					
0	0					

Now we fill in the missing entries in the table, working from left to right. The first three columns can be filled in by consulting the basic tables given above:

p	q	$p \Rightarrow q$	$\neg p$	$\neg q$	$\neg q \Rightarrow \neg p$	$(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$
1	1	1	0	0		
1	0	0	0	1		
0	1	1	1	0		
0	0	1	1	1		

Now the $\neg q \Rightarrow \neg p$ column can be filled in, using the values in the $\neg p$ and $\neg q$ columns and the basic table for $\Rightarrow$:

p	q	$p \Rightarrow q$	$\neg p$	$\neg q$	$\neg q \Rightarrow \neg p$	$(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$
1	1	1	0	0	1	
1	0	0	0	1	0	
0	1	1	1	0	1	
0	0	1	1	1	1	

Finally the last column can be filled in, using the values in the $p \Rightarrow q$ and $\neg q \Rightarrow \neg p$ columns and the basic table for $\Rightarrow$ again:

p	q	$p \Rightarrow q$	$\neg p$	$\neg q$	$\neg q \Rightarrow \neg p$	$(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$
1	1	1	0	0	1	1
1	0	0	0	1	0	1
0	1	1	1	0	1	1
0	0	1	1	1	1	1

The fact that every entry in the final column (corresponding to our target expression) is a **1** tells us that the expression is a *tautology*, that is, a expression which is true whatever the truth-values of its symbols p and q are. We will prove below that an expression is a tautology in this sense if and only if it is a theorem of **SL**. It follows then that the truth-table method is an algorithm for determining whether an expression is a theorem of **SL** or not, and thus that **SL** is decidable.

First, one more example. Let us ask if the connective $\Rightarrow$ is associative, that is, whether

$$((p \Rightarrow q) \Rightarrow r) \quad \Leftrightarrow \quad (p \Rightarrow (q \Rightarrow r))$$

is a theorem of **SL** or not. Here is the table:

p	q	r	$p \Rightarrow q$	$(p \Rightarrow q) \Rightarrow r$	$q \Rightarrow r$	$p \Rightarrow (q \Rightarrow r)$	The expression
1	1	1	1	1	1	1	1
1	1	0	1	0	0	0	1
1	0	1	0	1	1	1	1
1	0	0	0	1	1	1	1
0	1	1	1	1	1	1	1
0	1	0	1	0	0	1	0
0	0	1	1	1	1	1	1
0	0	0	1	0	1	1	0

First notice that the leftmost columns, corresponding to the letters p, q and r must contain every combination of truth-values. Consequently, if the expression being investigated has n letters, the table must have 2^n rows. Second, observe that som of the entries in the final column are not 1 in this case. From this we conclude that the expression $((p \Rightarrow q) \Rightarrow r) \quad \Leftrightarrow \quad (p \Rightarrow (q \Rightarrow r))$ is not a tautology, and hence not a theorem of **SL**.

We may re-interpret the columns of these tables as tables of functions. For example, take the table of $p \Rightarrow q$ above. Think of it as a table of a function of two variables (the values of p and q), each of which can take values from the set $\mathbf{B} = \{0, 1\}$. It is better not to confuse the expression $p \Rightarrow q$ with its truth-table function, so let us call the function $T_{p \Rightarrow q}$. It is thus a function $\mathbf{B}^2 \to \mathbf{B}$. Doing the same thing for the first big table above, we have another way of looking at it, as a table of values of the functions involved:

e	$T_{p\Rightarrow q}(e)$	$T_{\neg p}(e)$	$T_{\neg q}(e)$	$T_{\neg q\Rightarrow\neg p}(e)$	$T_{(p\Rightarrow q)\Rightarrow(\neg q\Rightarrow\neg p)}(e)$
(1,1)	1	0	0	1	1
(1,0)	0	0	1	0	1
(0,1)	1	1	0	1	1
(0,0)	1	1	1	1	1

This interpretation will be used in the remainder of this section to prove that the Truth-table Algorithm works as claimed.

G.4 Definition

Let us write $\mathbf{B}$ for the set $\{0,1\}$ so that $\mathbf{B}^k = \{(e_1,e_2,\ldots,e_k) : \text{each } e_i = 0 \text{ or } 1\}$. In $\mathbf{B}^k$ we will write $\mathbf{O}$ for the all-0 k-tuple and $\mathbf{I}$ for the all-1 one: $\mathbf{O} = (0,0,\ldots,0)$ and $\mathbf{I} = (1,1,\ldots,1)$.

We will be working with functions $\mathbf{B}^k \to \mathbf{B}$. We define operations on these functions corresponding to the connectives; if f and g are such functions, then $\neg f$, $f \wedge g$, $f \vee g$, $f \Rightarrow g$ and $f \Leftrightarrow g$ are defined as follows:

$$(\neg f)(\mathbf{e}) = \begin{cases} 1 & \text{if } f(\mathbf{e}) = 0\,; \\ 0 & \text{otherwise.} \end{cases}$$

$$(f \wedge g)(\mathbf{e}) = \begin{cases} 1 & \text{if } f(\mathbf{e}) = g(\mathbf{e}) = 1\,; \\ 0 & \text{otherwise.} \end{cases} \qquad (f \vee g)(\mathbf{e}) = \begin{cases} 0 & \text{if } f(\mathbf{e}) = g(\mathbf{e}) = 0\,; \\ 1 & \text{otherwise.} \end{cases}$$

$$(f \Rightarrow g)(\mathbf{e}) = \begin{cases} 0 & \text{if } f(\mathbf{e}) = 1 \text{ and } g(\mathbf{e}) = 0\,; \\ 1 & \text{otherwise.} \end{cases} \qquad (f \Leftrightarrow g)(\mathbf{e}) = \begin{cases} 1 & \text{if } f(\mathbf{e}) = g(\mathbf{e})\,; \\ 0 & \text{otherwise.} \end{cases}$$

To each expression C we define its *truth-table function*, $T_C : \mathbf{B}^k \to \mathbf{B}$ by induction over the construction of C, as follows:

► A.5

(i) If C is a proposition symbol p_i, then $T_C(e_1,e_2,\ldots,e_k) = e_i$.

(ii) If C is $\neg A$, then $T_C(\mathbf{e}) = \neg T_A(\mathbf{e})$ for all $\mathbf{e} \in \mathbf{B}^k$.

(iii) If C is $A \wedge B$, then $T_C(\mathbf{e}) = T_A(\mathbf{e}) \wedge T_B(\mathbf{e})$ for all $\mathbf{e} \in \mathbf{B}^k$.

(iv) If C is $A \vee B$, then $T_C(\mathbf{e}) = T_A(\mathbf{e}) \vee T_B(\mathbf{e})$ for all $\mathbf{e} \in \mathbf{B}^k$.

(v) If C is $A \Rightarrow B$, then $T_C(\mathbf{e}) = T_A(\mathbf{e}) \Rightarrow T_B(\mathbf{e})$ for all $\mathbf{e} \in \mathbf{B}^k$.

(vi) If C is $A \Leftrightarrow B$, then $T_C(\mathbf{e}) = T_A(\mathbf{e}) \Leftrightarrow T_B(\mathbf{e})$ for all $\mathbf{e} \in \mathbf{B}^k$.

Finally, we say that A is a *truth-table tautology* (TTT) if $T_A(\mathbf{e}) = 1$ for all $\mathbf{e} \in \mathbf{B}^k$. Note that all of this is just a complicated, but precise, way of describing the functions which are tabulated by a truth table, as described in the preceding section.

The remainder of this section is devoted to a proof of

G.5 Theorem

$$\vdash A \quad \text{if and only if} \quad A \text{ is a TTT.}$$

This theorem will establish the decidability of the theory, because whether a expression is a TTT or not is clearly decidable. (For each of the 2^k "vectors" $\mathbf{e}$, compute $T_A(\mathbf{e})$ according to the recipe given by their definition, and see if all the values are 1. The truth-table method, as described in G.4 is simply an organised way of doing this.) ▶ G.4

One half of the proof of this theorem is straightforward:

G.6 Half of the theorem

$$\text{If} \quad \vdash A \quad \text{then} \quad A \text{ is a TTT.}$$

Proof. A is a theorem if and only if it is one of the lines in a formal proof. Therefore it is enough to show that the three axioms are TTTs and that, if A and $A \Rightarrow B$ are TTTs, then so is B.

Consider Axiom A1, $A \Rightarrow (B \Rightarrow A)$. Let $\mathbf{e} \in \mathbf{B}^k$ and check all possible combinations of values of $T_A(\mathbf{e})$ and $T_B(\mathbf{e})$.

If $T_A(\mathbf{e}) = 1$ and $T_B(\mathbf{e}) = 1$ then $T_{B\Rightarrow A}(\mathbf{e}) = 1$ and so $T_{A\Rightarrow(B\Rightarrow A)}(\mathbf{e}) = 1$.

If $T_A(\mathbf{e}) = 1$ and $T_B(\mathbf{e}) = 0$ then $T_{B\Rightarrow A}(\mathbf{e}) = 1$ and so $T_{A\Rightarrow(B\Rightarrow A)}(\mathbf{e}) = 1$.

If $T_A(\mathbf{e}) = 0$ and $T_B(\mathbf{e}) = 1$ then $T_{B\Rightarrow A}(\mathbf{e}) = 0$ and so $T_{A\Rightarrow(B\Rightarrow A)}(\mathbf{e}) = 1$.

If $T_A(\mathbf{e}) = 0$ and $T_B(\mathbf{e}) = 0$ then $T_{B\Rightarrow A}(\mathbf{e}) = 1$ and so $T_{A\Rightarrow(B\Rightarrow A)}(\mathbf{e}) = 1$.

The proofs for Axioms A2 and A3 are the same, except that eight cases must be checked for each.

Now consider Modus Ponens. Suppose that A and $A \Rightarrow B$ are TTTs. Let $\mathbf{e} \in \mathbf{B}^k$. Then $T_A(\mathbf{e}) = 1$ (since A is a TTT) and $T_A(\mathbf{e}) = 1 \;\Rightarrow\; T_B(\mathbf{e}) = 1$ (by definition of $T_{A\Rightarrow B}$) and so $T_B(\mathbf{e}) = 1$ as required. ■

G.7 Definition

(i) A *conjunctive term* is a expression of the form $Q_1 \wedge Q_2 \wedge \ldots \wedge Q_k$ where, for each i, Q_i is either p_i or $\neg p_i$. (So every proposition symbol occurs exactly once, either with or without a preceding not-sign.)

(ii) A *disjunctive form* is a expression of one of the two forms

$$\mathbf{F} \qquad \text{or} \qquad C_1 \vee C_2 \vee \ldots \vee C_s \qquad \text{where } C_1, C_2, \ldots, C_s \text{ are distinct conjunctive terms.}$$

G.8 Notes

(1) We will assume that some fixed order has been chosen for the conjunctive terms in the second form of the expression. It does not matter what it is, so long as we choose one (say, lexicographic) and stick to it.

(2) The form **F** is used to act as the "gold-plated" always-false expression, such as $\neg(p_1 \Rightarrow p_1)$ (see Section F.26).

▶ F.26

(3) We will want to talk about "the conjunctive terms which occur in" any given disjunctive form. It is obvious what this means for the second form above. We will simply *define* **F** to contain no conjunctive terms.

(4) Having made these clarifications, we can now say that, given any set of conjunctive terms, there is exactly one disjunctive form which contains just these terms and no others.

G.9 Definition

To any member $\mathbf{e}$ of $\mathbf{B}^k$ there corresponds a conjunctive term $K_{\mathbf{e}} \ \ominus \ Q_1 \wedge Q_2 \wedge \ldots \wedge Q_k$. where

$$Q_i \ \ominus \ \begin{cases} p_i & \text{if } e_i = 1, \\ \neg p_i & \text{if } e_i = 0. \end{cases}$$

Then, for any expression A, we define its *disjunctive form* A_{DF} by

$$A_{\mathrm{DF}} \quad \ominus \quad C_1 \vee C_2 \vee \ldots \vee C_s$$

where $C_1, C_2, \ldots, C_s$ are just the conjunctive terms $K_{\mathbf{e}}$ for which $T_A(\mathbf{e}) = 1$, in our chosen order. (Note: in the case $\mathbf{e} = \mathbf{O}$, we interpret this to mean that A_{DF} is **F**.)

G.10 Lemma

(i) A conjunctive term $C \ \ominus \ K_{\mathbf{e}}$ satisfies

$$T_C(\mathbf{e}) = 1 \qquad \text{and} \qquad T_C(\mathbf{e}') = 0 \quad \text{for all } \mathbf{e}' \neq \mathbf{e}$$

(ii) If $D \ \ominus \ C_1 \vee C_2 \vee \ldots \vee C_s$ is a disjunctive form, then $T_D(\mathbf{e}) = 1$ for exactly those $\mathbf{e}$ for which $K_{\mathbf{e}}$ is one of the terms C_i of D.

Proof. **(i)** follows immediately from the definition and **(ii)** follows easily from **(i)**. ■

This lemma has several useful immediate corollaries:

G.11 Corollaries

(i) Any disjunctive form is its own disjunctive form, that is, if D is a disjunctive form, then $D_{\text{DF}} \Leftrightarrow D$.

(ii) $T_A = T_{A_{\text{DF}}}$ for all expressions A.

(iii) Suppose that D is a disjunctive form. Then D is a TTT if and only if it contains all 2^k conjunctive terms.

(iv) Distinct disjunctive forms have distinct truth-table functions.

G.12 Lemma

Let $A \Leftrightarrow C_1 \vee C_2 \vee \ldots \vee C_s$, where $C_1, C_2, \ldots, C_s$ are all the 2^k conjunctive terms. Then $\vdash A$.

Proof. We have $\vdash p_i \vee \neg p_i$ for each i (Theorem E20b). By Theorem F.7(c) then, ▶ F.7

$$\vdash \quad (p_1 \vee \neg p_1) \wedge (p_2 \vee \neg p_2) \ .$$

Now, by repeated applications of distributivity, commutativity and associativity (Theorems E15–E17),

$$\vdash \quad (p_1 \wedge p_2) \vee (p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge p_2) \vee (\neg p_1 \wedge \neg p_2) \ ,$$

which is the lemma in the case $k = 2$. If $k \geq 3$, we also have $\vdash p_3 \vee \neg p_3$ and so, by Theorem F.7(c) again, ▶ F.7

$$\vdash \quad \big((p_1 \wedge p_2) \vee (p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge p_2) \vee (\neg p_1 \wedge \neg p_2)\big) \ \wedge \ (p_3 \vee \neg p_3)$$

which, after umpteen applications of the distributive, associative and commutative laws expands out to the required eight-term expression. AND SO ON. ■

G.13 Corollary

Let D be a disjunctive form. Then $\vdash D$ if and only if it is a TTT.

G.14 Lemma

$$P_i \quad \vdash\!\!\!\vdash \quad C_1 \vee C_2 \vee \ldots \vee C_s \ ,$$

where $C_1, C_2, \ldots, C_s$ are all the 2^{k-1} conjunctive terms of the form $Q_1 \wedge Q_2 \wedge \ldots \wedge Q_k$ in which $Q_i = P_i$.

Proof. is the same as for the preceding lemma. ■

G.15 Lemma

Let $A \Ominus A_1 \vee A_2 \vee \ldots \vee A_s$ be a disjunctive form in which $A_1, A_2, \ldots, A_s$ are some of the available conjunctive terms and let $B \Ominus B_1 \vee B_2 \vee \ldots \vee B_t$ be another disjunctive form in which the conjunctive terms $B_1, B_2, \ldots, B_t$ are exactly the ones which are *not* involved in A. (In the special case in which $A_1, A_2, \ldots, A_s$ happens to be all the conjunctive terms, we interpret this to mean $B \Ominus \mathbf{F}$. And vice versa.) Then $\neg A \dashv\vdash B$.

Proof. We show first that $\neg A \vdash B$. By the Deduction Theorem, it is enough to show that $\vdash \neg A \Rightarrow B$. From the definition of the disjunction symbol, this is the same as $\vdash \neg\neg A \vee B$.
► F.3 Now by Theorem F.3 and Substitution of Equivalences, this is the same as proving that $\vdash A \vee B$. Because of the assumed form of A and B, this is the preceding lemma.

Next we prove that $B \vdash \neg A$. Again we ring some changes: by the Deduction Theorem it is enough to prove that $\vdash B \Rightarrow \neg A$. This is the same as $\vdash \neg\neg(B \Rightarrow \neg A)$ which, by the definition of conjunction, is the same as $\neg(B \wedge A)$, so this is what we will establish.

Now $B \wedge A$ is of the form $(B_1 \vee B_2 \vee \ldots \vee B_t) \wedge (A_1 \vee A_2 \vee \ldots \vee A_s)$ and so, by repeated use of the distributive, associative and commutative laws,

$$B \wedge A \dashv\vdash (B_1 \wedge A_1) \vee (B_1 \wedge A_2) \vee \ldots \vee (B_t \wedge A_s)\ , \tag{1}$$

the terms on the right being all expressions of the form $B_q \wedge C_p$ (for $q = 1, 2, \ldots, t$ and $p = 1, 2, \ldots, s$). In each such term, B_q and A_p are distinct and so there is at least one index i such that A_p contains p_i and B_q contains $\neg p_i$, or vice versa. Then, using commutativity, there is a expression E such that $B_q \wedge A_p \dashv\vdash p_i \wedge \neg p_i \wedge E$. From this and Theorem E22a we get $\neg(B_q \wedge A_p)$. Since this is true of every one of the terms in the expansion (1) above, repeated use of Theorem E22b gives $\vdash \neg(B \wedge A)$. ■

G.16 Lemma

Let $A \Ominus A_1 \vee A_2 \vee \ldots \vee A_s$ and $B \Ominus B_1 \vee B_2 \vee \ldots \vee B_t$ be disjunctive forms in which $A_1, A_2, \ldots, A_s$ is a subset of $B_1, B_2, \ldots, B_t$. Then $A \vdash B$.

Proof. If we write $E_1, E_2, \ldots, E_u$ for those conjunctive terms which appear in B but not in A, then using distributivity and associativity, $A \vee E_1 \vee E_2 \vee \ldots \vee E_u \dashv\vdash B$. By Theorem E7a, $A \vdash A \vee E_1 \vee E_2 \vee \ldots \vee E_u$ and we are done. ■

G.17 Lemma

For any expression A, $\quad A \dashv\vdash A_{\mathrm{DF}}$.

Proof. is by induction over the construction of A.

Suppose first that $A = p_i$, a proposition symbol. Now $T_A(\mathbf{e}) = e_i$. Therefore $A_{\mathrm{DF}} \Ominus A_1 \vee A_2 \vee \ldots \vee A_s$, where $A_1, A_2, \ldots, A_s$ are just the conjunctive terms of the form $Q_1 \wedge Q_2 \wedge \ldots \wedge Q_k$ in which $Q_i = p_i$. The result follows by Lemma F12.

Suppose next that $A \Subset \neg B$. Let $B_{\mathrm{DF}} \Subset B_1 \vee B_2 \vee \ldots \vee B_t$ and, by the inductive hypothesis, $B \vdash B_{\mathrm{DF}}$. Then

$$A \vdash \neg B_{\mathrm{DF}} \vdash \neg(B_1 \vee B_2 \vee \ldots \vee B_t) \vdash A_1 \vee A_2 \vee \ldots \vee A_s$$

where $A_1, A_2, \ldots, A_s$ are all the conjunctive terms which are not amongst $B_1, B_2, \ldots, B_t$ (the last step being by Lemma F13). Since $B_{\mathrm{DF}} \Subset B_1 \vee B_2 \vee \ldots \vee B_t$, $T_B(\mathbf{e}) = 1$ for just those $\mathbf{e}$ such that $K_{\mathbf{e}}$ is amongst $B_1, B_2, \ldots, B_t$. Thus $T_A(\mathbf{e}) = T_{\neg B}(\mathbf{e}) = 1$ for exactly those $\mathbf{e}$ such that $K_{\mathbf{e}}$ is *not* amongst $B_1, B_2, \ldots, B_t$, that is, for those $\mathbf{e}$ such that $K_{\mathbf{e}}$ is amongst $A_1, A_2, \ldots, A_s$. Therefore $A_{\mathrm{DF}} = A_1 \vee A_2 \vee \ldots \vee A_s$ and we have proved that $A \vdash A_{\mathrm{DF}}$ in this case.

Finally suppose that $A \Subset B \Rightarrow C$. Let $B_{\mathrm{DF}} = B_1 \vee B_2 \vee \ldots \vee B_t$ and $C_{\mathrm{DF}} = C_1 \vee C_2 \vee \ldots \vee C_u$. Then $A_{\mathrm{DF}} = A_1 \vee A_2 \vee \ldots \vee A_s$, where $A_1, A_2, \ldots, A_s$ are all conjunctive terms *except* for those which occur in B_{DF} but not in C_{DF} (in other words, all conjunctive terms which occur in C_{DF} or do not occur in B_{DF}). By the inductive hypothesis, $B \vdash B_{\mathrm{DF}}$ and $C \vdash C_{\mathrm{DF}}$ and so $B \Rightarrow C \vdash B_{\mathrm{DF}} \vdash C_{\mathrm{DF}}$, that is,

$$A \vdash (B_1 \vee B_2 \vee \ldots \vee B_t) \Rightarrow (C_1 \vee C_2 \vee \ldots \vee C_u)\,.$$

By the definition of disjunction,

$$(B_1 \vee B_2 \vee \ldots \vee B_t) \Rightarrow (C_1 \vee C_2 \vee \ldots \vee C_u) \vdash \neg(B_1 \vee B_2 \vee \ldots \vee B_t) \vee (C_1 \vee C_2 \vee \ldots \vee C_u)$$

and then, by Lemma F13,

$$(B_1 \vee B_2 \vee \ldots \vee B_t) \Rightarrow (C_1 \vee C_2 \vee \ldots \vee C_u) \vdash \neg(D_1 \vee D_2 \vee \ldots \vee D_v) \vee (C_1 \vee C_2 \vee \ldots \vee C_u)$$

where $D_1, D_2, \ldots, D_v$ is the set of conjunctive terms which do not occur amongst $B_1, B_2, \ldots, B_t$. We now have

$$A \vdash (D_1 \vee D_2 \vee \ldots \vee D_v) \vee (C_1 \vee C_2 \vee \ldots \vee C_u)$$

and, using commutativity and Theorem E21 to remove repetitions, this expression is $\vdash A_{\mathrm{DF}}$. ■

G.18 Theorem G.5 — second half

It remains to prove that, if A is a TTT then $\vdash A$.

Proof. If A is a TTT then, by Corollary F9(ii), so is A_{DF}. Then, by Corollary F11, $\vdash A_{\mathrm{DF}}$. Then, by Theorem F15, $\vdash A$. ■

G.19 Corollary

SL is consistent.

Proof. We must show that the theory contains no theorem of the form $\vdash A \wedge \neg A$. By the definition of a truth-table function, $T_{A \wedge \neg A}(\mathbf{e}) = 0$ for all $\mathbf{e}$ and so $A \wedge \neg A$ is not a TTT. The result follows. ■

G.20 Corollary

SL is not complete.

Proof. We must demonstrate the existence of a expression A such that neither A nor $\neg A$ is a theorem. This is so with $A \ominus p_1$ because

$$T_{p_1}(\mathbf{O}) = 0 \qquad \text{so } p_1 \text{ is not a TTT.}$$

$$T_{\neg p_1}(\mathbf{I}) = 0 \qquad \text{so } \neg p_1 \text{ is not a TTT.}$$

■

H Other axiomatisations

There are many other ways of setting up Sentential Logic as a deductive system. Here are a few.

H.1 Use of prefix notation

This is simply a different notation for expressions, one which allows parentheses to be done away with. It is more difficult for humans to read, probably easier for computers. Certain elementary facts, such as uniqueness of parsing, are more easily proved.

The symbols are the same, except that parentheses are not required:

Connectives	$\neg$	$\Rightarrow$		
Proposition symbols	p	q	r	and so on.

The rules for building expressions are

Any proposition symbol constitutes a expression.
If A is a expression, then so is $\neg A$.
If A and B are expressions then so is $\Rightarrow AB$.

The axioms are the same, translated into this notation:

$\Rightarrow A \Rightarrow BA$
$\Rightarrow \Rightarrow A \Rightarrow BC \Rightarrow \Rightarrow AB \Rightarrow AC$
$\Rightarrow \Rightarrow \neg A \neg\ B \Rightarrow \Rightarrow \neg ABA$

The only rule of inference is Modus Ponens, which now looks like this:

$$\frac{A\ ,\ \Rightarrow AB}{B}$$

H.2 An axiomatisation using only disjunction and negation

The symbols are:

Connectives	$\neg$	$\vee$		
Proposition symbols	p	q	r	and so on.
Punctuation	(	)		

The rules for building expressions are the obvious ones:

Any proposition symbol constitutes an expression.
If A is an expression, then so is $\neg A$.
If A and B are expressions then so is $(A \vee B)$.

In the semi-formal version of the language, we omit parentheses in the usual way and use $A \Rightarrow B$ as an abbreviation for $\neg A \vee B$. The axioms are:

(1) $A \vee A \Rightarrow A$

(2) $A \Rightarrow A \vee B$

(3) $A \vee B \Rightarrow B \vee A$

(4) $(B \Rightarrow C) \Rightarrow (A \vee B \Rightarrow A \vee C)$

As before, the only rule is Modus Ponens. Semi-formally, it looks the same; formally, it looks like this:

$$\frac{A\ ,\ (\neg A \vee B)}{B}$$

H.3 Formalising definitions

We can always make a symbol used as an abbreviation in the semi-formal version of the language a part of the formal language by adding it to the symbol list and adding an axiom or axioms sufficient to make it equivalent to its definition.

For example, if we start with our original system, defined at the beginning of this chapter, we can add the $\wedge$ symbol and the axioms

$A \wedge B \Rightarrow A$
$A \wedge B \Rightarrow B$
$A \Rightarrow (B \Rightarrow A \wedge B)$.

Alternatively, we can add the new symbol and some new rules of inference. For example, again starting from our original system, we can add the symbol $\wedge$ and two rules of inference:

$$\frac{\neg(A \Rightarrow \neg B)}{A \wedge B} \quad \text{and} \quad \frac{A \wedge B}{\neg(A \Rightarrow \neg B)}\ .$$

H.4 Minimal axiomatisations

It is possible to axiomatise Sentential Logic using the symbols $\neg$ and $\Rightarrow$ and only one axiom, to whit:

$$\Big[\big[\big((A \Rightarrow B) \Rightarrow (\neg C \Rightarrow \neg D)\big) \Rightarrow C\big] \Rightarrow E\Big] \ \Rightarrow\ \big[(E \Rightarrow A) \Rightarrow (D \Rightarrow A)\big]$$

(Here I have used several different types of parentheses to make the axiom easier to read.)

Even more stripped down is an axiomatisation using the *Scheffer stroke* or *nor* function (if we wanted to add it to ordinary logic, we would define it by $A|B \Leftrightarrow \neg(A \vee B)$) and a single axiom:

$$(A|(B|C)) \mid \Big[\big[D|(D|D)\big] \mid \big[(E|B) \mid \big((A|E)|(A|E)\big)\big]\Big] \ .$$

I Equivalent deductive systems

In the last section we considered several different axiomatisations of Sentential Logic. Assuming that we have checked that each of the axiom systems implies the other, it is more or less intuitively clear that these axiomatisations are equivalent. There are some details that need clearing up. Consider our original system, defined early in this chapter and the one in H.2 above (let us call them **SLa** and **SLb** for now). Each of the two axiomatisations contains a symbol that the other does not have. We can easily translate any expression in **SLa** into an expression in **SLb**, giving us a translation function $\varphi : \mathbf{SLa} \to \mathbf{SLb}$ defined inductively by

▶ H.2

$$\text{If } p \text{ is a proposition symbol, then } \varphi(p) \Leftrightarrow p\,,$$
$$\varphi(\neg A) \Leftrightarrow \neg\varphi(A) \qquad \text{and} \qquad \varphi(A \Rightarrow B) \Leftrightarrow \neg\varphi(A) \vee \varphi(B)\,.$$

In the same way, we have an obvious reverse translation function $\psi : \mathbf{SLb} \to \mathbf{SLa}$ defined inductively by

$$\text{If } p \text{ is a proposition symbol, then } \psi(p) \Leftrightarrow p\,.$$
$$\psi(\neg A) \Leftrightarrow \neg\psi(A) \qquad \text{and} \qquad \psi(A \vee B) \Leftrightarrow \neg\psi(A) \Rightarrow \psi(B)\,.$$

If these functions were inverse bijections, there would be nothing more to say: but they are not. For example, if p and q are two proposition symbols,

$$\psi(\varphi(p \Rightarrow q)) \Leftrightarrow \neg\neg p \Rightarrow q.$$

However we do observe (and expect) that, for any expression A in **SLa**, $\psi(\varphi(A)) \Leftrightarrow A$ and, for any expression B in **SLb**, $\varphi(\psi(B)) \Leftrightarrow B$.

This leads us to a general definition of equivalence of two deductive systems. We must allow for the fact that, if our languages do not contain logic (or even if they do), they may not contain the formal equivalence symbol $\Rightarrow$, so we use the relation $\dashv\vdash$ instead.

I.1 Definition

Let **A** and **B** be two formal systems. We will say that **A** and **B** are *equivalent* if there are functions $\varphi : \mathbf{A} \to \mathbf{B}$ and $\psi : \mathbf{B} \to \mathbf{A}$ (that is, to each expression A of **A** there is defined a corresponding expression $\varphi(A)$ of **B** and to each expression B of **B** there is defined a corresponding expression $\psi(B)$ of **A**) with these properties:

(1) For any expression A of **A**, $A \dashv\vdash \psi(\varphi(A))$ in **A**.

(1′) For any expression B of **B**, $B \dashv\vdash \varphi(\psi(B))$ in **B**.

(2) If $H_1, H_2, \ldots, H_k \vdash A$ in **A**, then $\varphi(H_1), \varphi(H_2), \ldots, \varphi(H_k) \vdash \varphi(A)$ in **B**.

(2′) If $K_1, K_2, \ldots, K_k \vdash B$ in **B**, then $\psi(K_1), \psi(K_2), \ldots, \psi(K_k) \vdash \psi(B)$ in **A**.

It is a straightforward exercise to prove that equivalence of formal systems, as just defined, is an equivalence relation in the ordinary sense.

I.2 Note

Part (2) of the theorem, in the case $k = 0$, tells us that if A is a theorem of **A**, then $\varphi(A)$ is a theorem of **B**. In the same way, if B is a theorem of **B**, then $\psi(B)$ is a theorem of **B**.

I.3 Theorem

Let **A** and **B** be two formal systems. Then **A** and **B** are *equivalent* if and only if there are functions $\varphi : \mathbf{A} \to \mathbf{B}$ and $\psi : \mathbf{B} \to \mathbf{A}$ with these properties:

(1) For any expression A of **A**, $A \vdash\!\dashv \psi(\varphi(A))$ in **A**.

(1′) For any expression B of **B**, $B \vdash\!\dashv \varphi(\psi(B))$ in **B**.

(2) For any axiom A of **A**, $\vdash \varphi(A)$ in **B**.

(2′) For any axiom B of **B**, $\vdash \psi(B)$ in **A**.

(3) If $\frac{H_1, H_2, \ldots, H_k}{A}$ is any rule of **A**, then $\varphi(H_1), \varphi(H_2), \ldots, \varphi(H_k) \vdash \varphi(A)$ in **B**.

(3′) If $\frac{K_1, K_2, \ldots, K_k}{B}$ is any rule of **B**, then $\psi(K_1), \psi(K_2), \ldots, \psi(K_k) \vdash \psi(B)$ in **A**.

Proof. Suppose first the four conditions of the definition hold. Then we are given that Conditions (1) and (1′) of this theorem hold. Also Conditions (2) and (2′) of this theorem hold because any axiom is automatically a theorem. Similarly Condition (3) of this theorem holds because if $\frac{H_1, H_2, \ldots, H_n}{A}$ is a rule of **A**, then $H_1, H_2, \ldots, H_n \vdash A$ in **A**. Condition (3′) holds for the same reason.

Now suppose that the six conditions of this theorem hold. As before, we are given that Conditions (1) and (1′) of the Definition hold. We now prove that Condition (2) of the definition holds.

Suppose that $H_1, H_2, \ldots, H_n \vdash A$ in **A**; let $L_1, L_2, \ldots, L_n$ be a proof of this deduction. Then every line L_i of this proof is either an axiom of **A** or follows from earlier lines by one of the rules of **A**. From Conditions (2) and (3) of this theorem then, every $\varphi(L_i)$ is either a theorem of **B** or follows from φ(those earlier lines) by a deduction in **B**. Then, using derived rules D.2 and D.3, it follows that $\varphi(L_1), \varphi(L_2), \ldots, \varphi(L_n)$ is a semi-formal proof of $\varphi(H_1), \varphi(H_2), \ldots, \varphi(H_k) \vdash \varphi(A)$ in **B**. ▶ D.2 ▶ D.3

The proof that Condition (2′) of the definition holds is the same. ■

I.4 Remark

This theorem looks more complicated than the definition, however it is useful because its conditions are usually easier to check than those of the definition. In particular, for many systems based on logic, the rules are the same for both systems and only the axioms differ; in this case Conditions (3) and (3′) hold automatically.

I.5 Exercise

Prove that the two versions of Sentential Logic just discussed (**SLa** and **SLb**) are indeed equivalent deductive systems. Note that you cannot assume that the Deduction Theorem holds in **SLb** until it has been proved.

3. PREDICATE LOGIC AND FIRST ORDER THEORIES

A A language for first order theories

A.1 Introduction

We now extend our languages and theories to deal with quantification. Consider for example the statement

$$\text{for all } x, y \text{ and } z, \text{ if } \quad x + y = x + z \quad \text{then} \quad y = z$$

which we might expect to be a theorem in a theory about the natural numbers. Expressing it more symbolically,

$$(\forall x)(\forall y)(\forall z)(x + y = x + z \;\Rightarrow\; y = z) \; .$$

Consider the component parts of this. First there are the variable symbols, x, y and z. Then there is the function symbol $+$ representing addition, a binary function (that is, a function of two variables). We wish to set up a general theory, so we will avoid infix and other special types of function notation; we make the expression look more generic by writing $\alpha(x, y)$ instead of $x + y$, and the expression becomes

$$(\forall x)(\forall y)(\forall z)(\alpha(x, y) = \alpha(x, z) \;\Rightarrow\; y = z) \; .$$

Or why change symbols? Write it

$$(\forall x)(\forall y)(\forall z)(+(x, y) = +(x, z) \;\Rightarrow\; y = z) \; .$$

The statement contains a relation, equality, also binary. We make the expression look even more generic by writing $=(x, y)$ for $x = y$, and our expression becomes

$$(\forall x)(\forall y)(=(+(x, y), +(x, z)) \;\Rightarrow =(y, z)) \; .$$

Here we have the essential components of a *first order theory* (that is, any theory based upon Predicate Logic): variable symbols, function and relation symbols, and the logical connectives augmented by quantifiers.

When discussing a first order theory, we usually have a particular interpretation in mind, such as Number Theory, Set Theory, Group Theory or whatever. When we do this we will have two kinds of axioms:

- Predicate Logic axioms, which are there to make the logic work and which are common to all first order theories. An example is

$$(\forall x)(P \Rightarrow Q) \;\Rightarrow\; ((\forall x)P \Rightarrow (\forall x)Q) \; .$$

- *Proper* axioms, specific to the particular theory, usually designed to encapsulate the properties of the structure being described. For instance, for elementary Number Theory we might have (reverting to semi-formal notation)

$$(\forall x)(x + 0 = x) \ .$$

There are several different ways we can think about a first order theory.

(1) We might have a specific interpretation in mind, such as elementary Number Theory, Group Theory or even one of the axiomatisations of Set Theory (which is powerful enough to contain all of mathematics) — the main ones are Von Neumann Bernays Gödel and Zermelo Fraenkel. In this case we will have a (usually small) number of functions and relations, each of which has a definite interpretation in our target structure (such as addition, equality, set-membership etc.). We will discuss several specific theories in Chapter 4.

(2) We might want to discuss first order theories in general, for instance in order to discover properties common to them all. Then we will have functions and relations, but no specific interpretation in mind.

(3) We might want to consider Predicate Logic itself, in much the same way as we considered Sentential Logic in the previous chapter. Then we will have no proper axioms at all. Predicate Logic is not uniquely defined — one can create different predicate logics by making different choices of the sets of function and relation symbols. However it turns out that the difference between different predicate logics is entirely inessential, in that, for any property of theories we are actually interested in, if it is true for one predicate logic then it is true for all of them. Therefore there is no harm in choosing one gold plated example as a representative and calling it Predicate Logic (with capitals) or just **PL**. This is what we will do.

Most of the important properties of **PL** are properties of all first order theories. There is, however, one important theorem about **PL** in this chapter which is not shared by all first order theories: it is consistent.

A.2 Symbols and strings

There are several categories of symbols:

The *connective symbols*	$\neg \quad \Rightarrow \quad \forall$	
The *variable symbols*	$x \quad y \quad z$	and so on.
The *function symbols*	See notes below.	
The *relation symbols*	See notes below.	
The *punctuation symbols*	$(\quad) \quad ,$	

A.3 Comments

(i) When we come to *specific* First Order Theories, the function symbols will be meant to have permanently assigned meanings. In ordinary mathematics, examples might be symbols such as $+$ or $\cap$. In an example such as "let f be a differentiable function ..." we would not include the letter f among the function symbols above — other arrangements are made for this usage (that is, functions created on the fly). The actual list of function symbols would depend on which particular language we are considering. In the same way, the relation symbols also will be meant to have permanently assigned meanings. In ordinary mathematics, examples would be $=$ or $\in$. Again, the actual list of relation symbols would depend on the particular example of a language we were considering. In a first order theory there would usually be only a finite number of function and relation symbols (though this is not necessary); in some systems one or other of these sets might even be empty.

In pure Predicate Logic, the function and relation symbols are not supposed to have any specific meanings, and we usually use non-specific symbols such as f and r to stand for them. It is usual to suppose that we have a finite (or at most denumerably infinite) collection of each, though this is not necessary. It is always supposed that the symbols have been chosen so that variable, function and relation symbols can be recognised as such.

(ii) Function symbols have an *arity*, which is a natural number (≥ 0) — a function of arity n is simply a function of n variables. (A function of arity n is called n-ary. Common terminology is *nullary* for 0-ary, *unary* for 1-ary, *binary* for 2-ary and *ternary* for 3-ary.) Note that nullary functions are just constants, so this allows us to pick out permanent names for important constants in our system, examples in ordinary mathematics being 0 and $\varnothing$.

Relation symbols also have arity. A nullary relation represents a truth value that does not depend on any variables, so nullary relations have the same role in Predicate Logic as proposition symbols have in Sentential Logic.

The arity of a function or relation symbol is also meant to be permanently attached to it and be instantly recognisable from the symbol itself. We will use the terms *function domain* and *relation domain* to refer to such a set of function or relation symbols with prescribed arities. The function domain and relation domain, taken together, are often referred to as the *signature* of the language.

(iii) Other connectives could be added to our list. Conspicuous by their absence are $\wedge$, $\vee$, $\Leftrightarrow$ and $\exists$. These connectives can be defined in terms of the ones given above, and so are redundant.

$$
\begin{array}{lcl}
P \vee Q & \circledminus & \neg P \Rightarrow Q \\
P \wedge Q & \circledminus & \neg(P \Rightarrow \neg Q) \\
P \Leftrightarrow Q & \circledminus & \neg((P \Rightarrow Q) \Rightarrow \neg(Q \Rightarrow P)) \\
(\exists x)P(x) & \circledminus & \neg(\forall x)\neg P(x)\ .
\end{array}
$$

We could cut down on the number of variable symbols by using a single letter, x say, and a prime. This would give us a potentially infinite number of variables, x , x' , x'' , x''' and so on. The same of course could be done to the function and relation symbols. We do not gain any useful simplicity from this, so we don't bother.

A.4 Definition: Terms

A string is a *term* if it is of one of the forms

(T1) A variable symbol.

(T2) $f(s_1, s_2, \ldots, s_n)$, where f is an n-ary function symbol, $n \geq 1$ and $s_1, s_2, \ldots, s_n$ are terms, or simply f in the case where f is a nullary function symbol.

A.5 Comments

(i) Think of terms as behaving like nouns and noun phrases in natural languages: they name things.

(ii) As already mentioned, a special case is a nullary function, which can be thought of as a constant. Examples are 0 (in a language for the integers) and $\varnothing$ (in a language for set theory).

A.6 Definition: Expressions (wffs)

A string is an *expression* (or *well-formed formula* or *wff*) if it is of one of the forms

(E1) $r(t_1, t_2, \ldots, t_n)$ where r is an n-ary relation symbol, $n \geq 1$ and $t_1, t_2, \ldots, t_n$ are terms, or simply r in the case were r is a nullary relation symbol.

(E2) $(\neg P)$ where P is an expression,

(E3) $(P \Rightarrow Q)$ where P and Q are expressions, or

(E4) $(\forall x P)$ where x is a variable symbol and P is an expression.

A.7 Comments

(i) Expressions of the form (E1) are called *atomic*.

(ii) Expressions normally represent sentences (e.g. $2 + 2 = 4$), which are either true or false, or predicates (e.g. $x + y = z$), whose truth or falsity depends upon what the variables represent.

(iii) Again, we will make the usual changes in our semi-formal language to improve readability. Superfluous parentheses will be removed, and quantifiers will usually be written $(\forall x)P$.

(iv) What about the other connectives? Using the equivalences listed in Section A2, we can consider the connectives $\wedge$, $\vee$, $\Leftrightarrow$ and $\exists$ to be simply convenient abbreviations, in our semi-formal language, for more complicated expressions in the formal one.

(v) If we want to include the symbols **T** and **F** (for true and false), they are nullary relations, and so are expressions.

Again we have a *unique parsing* lemma.

A.8 Lemma

(i) A string cannot be both a term and an expression.

(ii) Let A be a term. Then only one of cases (T1) or (T2) of the definition above hold. If (T1) holds, there is a unique variable symbol x such that $A \ominus x$. If (T2) holds, then there is a unique function symbol f and terms $s_1, s_2, \ldots, s_n$ such that $A \ominus f(s_1, s_2, \ldots, s_n)$.

(iii) Let A be an expression. Then only one of cases (E1) to (E4) of the definition above hold. If (E1) holds, then there is a unique relation symbol r and terms $t_1, t_2, \ldots, t_n$ such that $A \ominus r(t_1, t_2, \ldots, t_n)$. If (E2) holds, then A is unique, if (E3) holds then P and Q are unique and if (E4) holds then x and P are unique.

A.9 Definition: Binding

Let P be an expression and consider some occurrence of the variable symbol x in P. Then x is *bound* in P if it is part of some occurrence of an expression of the form $(\forall x Q)$ in P. Otherwise it is *free*.

If we want to be careful, we can define this by induction over the construction of P, using Definition A3:

- Every occurrence of any variable in an atomic expression $r(t_1, t_2, \ldots, t_n)$ is free.
- An occurrence of a variable in $\neg P$ is free or bound according as it is free or bound in P.
- An occurrence of a variable in $P \Rightarrow Q$ is free or bound according as it is free or bound in P or Q (whichever it occurs in).
- All occurrences of a variable x are bound in $(\forall x P)$; any occurrence of any other variable is free or bound according as it is free or bound in P.

A.10 Definition: Sentences

A *sentence* is an expression which contains no free occurrences of any variables. It is also called a *closed expression*.

A.11 Definition: Substitution

Let P be any expression, x a variable and t a term. We define $P[x/t]$, the *result of substituting* t *for* x *in* P to be the expression formed from P by replacing every free occurrence of x by t.

(It is a trivial matter to verify that the result of such a substitution is always an expression.)

This substitution is *acceptable* provided that there is no variable symbol y which occurs in t and such that some free occurrence of x in P is within the scope of a $(\forall y)$ quantifier in P.

A common special case is when one variable is being substituted for another. If x and y are variable symbols, then the substitution $[x/y]$ is acceptable in P provided no free occurrence of x in P is within the scope of a $(\forall y)$ quantifier.

A careful definition of all this, by induction over the construction of P, goes as follows.

First we define substitution into a term s.

- If s is a variable symbol, there are two possibilities,
  (i) $s \ominus x$, in which case $s[x/y] \ominus x[x/y] \ominus y$;
  (ii) $s \not\ominus x$, in which case $s[x/y] \ominus s$
- If s is $f(s_1, s_2, \ldots, s_n)$, where f is a function symbol and $s_1, s_2, \ldots, s_n$ are terms, then

$$s[x/t] \quad \ominus \quad f(s_1[x/t], s_2[x/t], \ldots, s_n[x/t]) \ .$$

All substitutions in terms are acceptable.

Now we define substitutions in expressions.

- If P is $r(s_1, s_2, \ldots, s_n)$, where r is a relation symbol and $s_1, s_2, \ldots, s_n$ are terms, then

$$P[x/t] \quad \ominus \quad r(s_1[x/t], s_2[x/t], \ldots, s_n[x/t]) \ .$$

  This is always acceptable.
- If P is $\neg Q$, then $P[x/t]$ is $\neg Q[x/t]$. The substitution $[x/y]$ is acceptable in P if it is acceptable in Q.
- If P is $(Q \Rightarrow R)$, then $P[x/t]$ is $(Q[x/t] \Rightarrow R[x/t])$. The substitution $[x/y]$ is acceptable in P if it is acceptable in both Q and R.
- If P is $(\forall y Q)$, where $y \not\ominus x$, then $P[x/t]$ is $(\forall y Q[x/t])$. The substitution $[x/y]$ is acceptable in P if it is acceptable in Q *and* the variable y does not occur in t.
- If P is $(\forall x Q)$, then $P[x/t]$ is P. (Note that, if the variable x does occur in Q, then it is left unchanged, because it is bound.) This substitution is always acceptable.

▶ I

In Section I at the end of this chapter there is some discussion of the ideas of binding and acceptability.

A.12 Comments

(i) In our semiformal language, we often write $P(x)$ for an expression in which the variable x may or may not occur free, and then write $P(t)$ for the result of substituting t for x in P(that is, to mean $P[x/t]$). This slightly looser notation is more in line with normal usage. It is often easier to read but needs to be handled with care because it can be ambiguous.

(ii) A couple of trivial cases of substitution will come up occasionally: firstly the substitution $[x/x]$ is always acceptable and $P[x/x] \ominus P$. Secondly, if the variable symbol x does not occur anywhere in the expression P, then $[x/y]$ is acceptable in P and $P[x/y] \ominus P$.

A.13 Axioms and rules for Predicate Logic

These axioms are schema: P , Q and R stand for any expressions, x for any variable symbol and t for any term.

(PL1) $P \Rightarrow (Q \Rightarrow P)$

(PL2) $(P \Rightarrow (Q \Rightarrow R)) \;\Rightarrow\; ((P \Rightarrow Q) \Rightarrow (P \Rightarrow R))$

(PL3) $(\neg P \Rightarrow \neg Q) \;\Rightarrow\; ((\neg P \Rightarrow Q) \Rightarrow P)$

(PL4) $(\forall x)(P \Rightarrow Q) \;\Rightarrow\; ((\forall x)P \;\Rightarrow\; (\forall x)Q)$

(PL5) $P \Rightarrow (\forall x)P$ provided that x does not occur free in P.

(PL6) $(\forall x)P \Rightarrow P[x/t]$ provided that the substitution $[x/t]$ is acceptable in P.

The Axiom PL6 is called *Particularisation.*

There are two rules:

Modus Ponens (MP) $\dfrac{P \;,\; P \Rightarrow Q}{Q}$

Universal generalisation (UG) $\dfrac{P}{(\forall y)P[x/y]}$

where

(i) the substitution $[x/y]$ is acceptable in P

and, if this rule is used in the proof of a deduction, there are two further provisos:

(ii) if x occurs free in P, then it must not occur free in any of the hypotheses and

(iii) if y occurs free in P, then it must not occur free in any of the hypotheses.

In an application of UG, we will call x the variable *being generalised.*

Notation

In the alternative notation described above, the new axioms look like this:

(PL4) $(\forall x)(P(x) \Rightarrow Q(x)) \;\Rightarrow\; ((\forall x)P(x) \;\Rightarrow\; (\forall x)Q(x))$

(PL5) $P \Rightarrow (\forall x)P$ provided that x does not occur free in P.

(PL6) $(\forall x)P(x) \Rightarrow P(t)$ provided that the substitution $[x/t]$ is acceptable in P.

Universal generalisation looks like this: $\dfrac{P(x)}{(\forall y)P(y)}$.

A.14 Comments

(i) In some cases the statement of UG simplifies a bit. Firstly, in a proof of a theorem there are no hypotheses and so Proviso (ii) (that neither x nor y occur free in any hypotheses)

to UG can be ignored. The acceptablility condition however must always hold.

Secondly, quite often we are not interested in changing the variable symbol. In this case UG look like this:

$$\frac{P}{(\forall x)P}$$

In this case Proviso (i) can be ignored, because we are effectively making the substitution $[x/x]$ which is always acceptable and Proviso (ii) simplifies in the obvious way.

(ii) If, in a proof of P from hypotheses $H_1, H_2, \ldots, H_k$, UG is used on variable symbol x, then x must not occur free in any of $H_1, H_2, \ldots, H_k$. If now we add another (non-essential) hypothesis H_{k+1} which does happen to contain x free, the deduction of P from $H_1, H_2, \ldots, H_k, H_{k+1}$ is no longer valid. It seems paradoxical that adding extra hypotheses can invalidate a proof, but this is so.

This could be a real nuisance in developing proofs, however all is not lost. While adding the new hypothesis H_{k+1} may invalidate the *given* proof of $H_1, H_2, \ldots, H_k \vdash P$, we will see that there is always *a* proof, perhaps different, of $H_1, H_2, \ldots, H_k, H_{k+1} \vdash P$ (this is the corollary to Theorem C.4 below).

► C.4

(iii) In a first order system, where further axioms may be added to the six listed above, it is not usually expected that they be given as schemata (though there is no harm in doing this). On the other hand, it is expected that they be sentences (closed expressions). Suppose, for example, we considering whether to use $x + 0 = x$ or $(\forall x)(x + 0 = x)$ as an axiom for elementary number theory. One might think that there is not much to choose between them because we can easily prove them equivalent, using PL6 in one direction and UG in the other. Given the quantified version one can always conclude the unquantified one by PL6, however in the context of a larger proof the passage from the unquantified version to the quantified one using UG might fail, for the reasons mentioned in the previous item. Therefore one should always use fully quantified axioms.

(iv) BEWARE OF AXIOM PL5! It can only be used when P does *not* contain x free. This means that it is of very limited usefulness, for instance in proving weird things like

$$\begin{aligned} 2 + 2 = 4 \quad &\Rightarrow \quad (\forall x)(2 + 2 = 4) \\ (\forall x)P \quad &\Rightarrow \quad (\forall x)(\forall x)P \ . \end{aligned}$$

B The Deduction Theorem

The Deduction Theorem needs to be proved again in this new context. In order to do this we must observe that Theorem **2**.B.1 and Deduction **2**.B.3 of Chapter **2** still hold good for Predicate Logic: simply look at the proofs given there and observe that they obey the rules for a proof in Predicate Logic.

▶ **2**.B.1
▶ **2**.B.3

B.1 The Deduction Theorem for PL

If $H_1, H_2, \ldots, H_k, P \vdash Q$ then $H_1, H_2, \ldots, H_k \vdash P \Rightarrow Q$.

Proof. The proof is exactly the same as before, except that we must check one extra case: we are dealing with a line, L say, in the old proof which is the result of applying UG, and we want to deduce $P \Rightarrow L$ in the new one.

Our assumption is that L is of the form $(\forall x)R(x)$ and there was a previous line of the form $R(x)$ in the old proof. Also, because the old proof was valid, x does not occur free in any of the hypotheses $H_1, H_2, \ldots, H_k, P$.

In our new proof then, we already have a line of the form $P \Rightarrow R(x)$ and we wish to show that $P \Rightarrow (\forall x)R(x)$ can be deduced. Insert extra lines as follows:

1.	$P \Rightarrow R(x)$	We know that this is there already.
2.	$(\forall x)(P \Rightarrow R(x))$	UG. Note that the proviso holds.
3.	$(\forall x)P \Rightarrow (\forall x)R(x)$	2 and Axiom PL4
4.	$P \Rightarrow (\forall x)P$	Axiom PL5
5.	$P \Rightarrow (\forall x)R(x)$	Theorem B.3 of Sentential Logic applied to 3 and 4.

(Note that it is OK to use Theorem B.3 of Sentential Logic in this proof, because the proof of that given in Chapter 2 did not use the (old) Deduction Theorem.) ■

C Using theorems and deductions in proofs

In Sentential Logic, as part of our semiformal theory, we can always drop a theorem as a line into any proof. As a recipe for formalising this action, simply insert before this line its full formal proof.

Also in Sentential Logic we allowed deductions to be used in any proof. If lines $H_1, H_2, \ldots, H_k$ already appear in a proof and the deduction $H_1, H_2, \ldots, H_k \vdash P$ has already been proved, then we may now insert the line P. Now Note (ii) at the end of A7 above raises the spectre that this may no longer work for Predicate Logic. To cut a long story short, it does work (See C.4 below), but the proof is more tricky and will require some preparation.

C.1 Remarks about repeated substitution

At first sight you might expect that substituting y for x in an expression (or term) and then following that by substituting z for y in the result would be the same as just substituting z for x in the first place. In symbols

$$X[x/y][y/z] \ominus X[x/z] \qquad ??$$

In fact, this does not always work. A simple example shows the sort of thing that can go wrong: let $x + y = 3$ be our expression. Then

$$\begin{aligned} & (x+y=3)[x/y] \ominus y+y=3 \\ \text{and so} \quad & (x+y=3)[x/y][y/z] \ominus z+z=3 \\ \text{whereas} \quad & (x+y=3)[x/z] \ominus z+y=3\,. \end{aligned}$$

In the same way, we might expect that substituting y for x and then x for y should get you back to the expression you started with, but again no:

$$\begin{aligned} & (x+y=3)[x/y] \ominus y+y=3 \\ \text{and so} \quad & (x+y=3)[x/y][y/x] \ominus x+x=3\,. \end{aligned}$$

However all is not lost. Provided the intermediate variable symbol is *fresh*, that is, one which does not occur free in the expression you start with, then things work as (probably) planned.

The principle here is: provided that the variable symbol u does not occur free in X, then

$$X[x/u][u/y] \ominus X[x,y] \tag{–1}$$

and

$$X[x/u][u/x] \ominus X\,. \tag{–2}$$

We will have one occasion below (at the end of the next lemma) to make such a repeated substitution when the intermediate variable symbol u does indeed occur free in X. Consider $X[x/u][u/y]$. Here all the variable symbols which were x get changed to u and then to y and all the variable symbols which were u in the first place just get changed to y. Thus this is the same as directly changing both u and x to y throughout:

$$X[x/u][u/y] \ominus X[x/y][u/y]\,. \tag{–3}$$

C.2 Preliminaries

(i) In proofs of theorems and deductions one often has what I will call *internal* variable symbols; these are ones which do not appear free in any of the hypotheses. To be boringly precise, suppose we have a proof $S_1, S_2, \ldots, S_n$ of a deduction $H_1, H_2, \ldots, H_h \vdash A$ (so that $A \ominus S_n$). Then the internal variable symbols here are any which occur free in any of $S_1, S_2, \ldots, S_n$ but not in any of $H_1, H_2, \ldots, H_h$.

(ii) In an application of Universal Generalisation,

$$\frac{P}{(\forall y)P[x/y]}$$

we will call the variable symbols x and y the *affected* variables, provided they are free in P.

Thus we can say that the rule above is a valid application of UG provided $[x/y]$ is acceptable in P and any affected variables in it are internal.

We will now show that internal variables can be changed to fresh ones more or less with impunity.

C.3 Lemma

Let

$$H_1, H_2, \ldots, H_h \vdash A \tag{–1}$$

be a deduction with proof

$$S_1, S_2, \ldots, S_n \tag{–2}$$

(so that $A \ominus S_n$).

(i) Let z be a variable symbol in this proof which is internal, in the sense that it does not occur free in any of the hypotheses $H_1, H_2, \ldots, H_h$, and let u be a *fresh* variable, in the sense that it does not occur anywhere, free or bound, in (–1) or (–2). Then substituting u for z throughout the proof to get

$$S'_1, S'_2, \ldots, S'_n\,,$$

where $S'_i \ominus S_i[z/u]$ for $i = 1.2. \ldots, n$, and placing the two sequences one after another thus

$$S'_1, S'_2, \ldots, S'_n, S_1, S_2, \ldots, S_n\,, \tag{–3}$$

all the steps are valid proof-steps in a proof of

$$H_1, H_2, \ldots, H_h \vdash S_n\,. \tag{–4}$$

in which the variable symbol z is never an affected variable in any UG step.

(ii) One can do this as many times as one likes:–

Let $z_1, z_2, \ldots, z_m$ be m distinct variable symbols in this proof which are internal, in the sense that none of them occur free in any of the hypotheses $H_1, H_2, \ldots, H_h$, and let $u_1, u_2, \ldots, u_m$ be an equal number of distinct fresh variables. Then substituting the u_i for the z_i throughout the proof to get

$$S'_1, S'_2, \ldots, S'_n$$

where, for $i = 1, 2, \ldots, n$,

$$S'_i \Subset S_i[z_1/u_1][z_2/u_2]\ldots[z_m/u_m]\,,$$

and placing the two sequences one after another thus

$$S'_1, S'_2, \ldots, S'_n, S_1, S_2, \ldots, S_n\,, \tag{–5}$$

all the steps are valid proof-steps in a proof of

$$H_1, H_2, \ldots, H_h \vdash S_n\,. \tag{–6}$$

in which none of the variable symbols $z_1, z_2, \ldots, z_m$ are affected variables in any UG step.

Proof (i). We check that, for each i, S'_i and S_i are valid steps in a proof of (–4) and that, if any such step is justified by UG, then z is not an affected variable.

If S_i is one of the axioms PL1, PL2 or PL3, then S'_i is just another instance of the same axiom. If S_i is any other axiom, then it is fully quantified, so S'_i is the same as S_i and is an axiom.

If S_i is one of the hypotheses, then z does not occur free in S_i, so S'_i is that same hypothesis.

If S_i follows by Modus Ponens from two earlier steps, S_j and $S_j \Rightarrow S_i$ say, then in (–4) we have earlier steps S'_j and $(S'_j \Rightarrow S'_i)$ and so S'_i follows by Modus Ponens also.

Finally, suppose that S_i follows fom a previous step by Universal Generalisation. Then

$$S_i \Subset (\forall y)S_j[x/y]\,, \tag{–7}$$

where S_j is an earlier step, the substitution $[x/y]$ is acceptable in S_j, if x is free in S_j then it is internal and similarly if y is free in S_j then it is internal.

In what follows it will be useful to observe that, since $S'_i \Subset S_i[z/u]$ and $S'_j \Subset S_j[z/u]$ by definition, with u fresh, we have also

$$S_i \Subset S'_i[u/z] \qquad \text{and} \qquad S_i \Subset S'_i[u/z] \tag{–8}$$

We need now to deal with a hierarchy of cases and subcases. We start with the possibilities that x and y might be the same symbol or different ones.

Case 1, x and y are the same symbol.

Then we have

$$S_i \Subset (\forall x)S_j \tag{–9a}$$

$$\text{and so} \qquad S'_i \Subset ((\forall x)S_j)[z/u] \tag{–9b}$$

where, if x is free in S_j, then it is internal. Now x might or might not be free in S_j, so we have subcases.

If x is not free in S_j this step has an alternative justification by PL5 and MP, which have been dealt with above. So now we assume that x is free in S_j.

There is now the possibility that x and z might or might not be the same symbol.

Case 1.1, x and z are different.

In this case (using the definition of substitution), (–9b) becomes

$$\begin{aligned} S'_i &\ominus (\forall x)(S_j[z/u]) \\ &\ominus (\forall x)S'_j \end{aligned}$$

and x is internal, so this is a valid use of UG. Also

$$S_i \ominus S'_i[u/z] \ominus (\forall x)S'_j[u/x]$$

and here x is internal and u is fresh, so this is also a valid use of UG. Note here that neither of the affected variables, u and x are z.

Case 1.2, x and z are the same.

In this case (–9a) and (–9b) become

$$\begin{aligned} & S_i \ominus (\forall z)S_j \\ \text{and so} \quad & S'_i \ominus ((\forall z)S_j)[z/u] \ominus (\forall z)S_j\,, \end{aligned}$$

the last step being because z is not free in $(\forall z)S_j$. But this gives $S'_i \ominus S_i \ominus (\forall x)(S'_j[u/z])$ and this is a valid use of UG since u is fresh and z is not affected, since it is not free in S'_j. Here the only affected variable is u, which is not z.

Case 2, x and y are different symbols.

We now have

$$\begin{aligned} & S_i \ominus (\forall y)(S_j[x/y]) && \text{(–10a)} \\ \text{and so} \quad & S'_i \ominus ((\forall y)(S_j[x/y]))[z/u] && \text{(–10b)} \end{aligned}$$

where x and y are different and, if x is free in S_j, then it is internal and the same is true of y.

If x is not free in S_j, then $S_i \ominus (\forall y)S_j$, so this is the same as Case 1, with y in place of x. So from now on we assume that x occurs free in S_j. Since (–10a) is a valid UG in the original proof, x is internal.

Now we have three subcass to consider, depending upon whether z is the same as Yx, the same as y or different from both.

Case 2.1, z is different from both x and y.

We already know that (–10a) is a valid UG and, for this case, both the affected variables, Yx and y, are different from z. Also

$$\begin{aligned} S'_i &\ominus S_i[z/u] \ominus ((\forall y)S_j[x/y])[z/u] && \text{by (–10a)} \\ &\ominus ((\forall y)(S_j[x/y][z/u]) && \text{since } z \notin y \\ &\ominus ((\forall y)(S_j[[z/u][x/y]) && \text{since } z \notin x, y \\ &\ominus (\forall y)S'_j[x/y] && \text{by definition of } S'_j\,. \qquad \text{(–11)} \end{aligned}$$

Now, if y is free in S'_j then it is also free in $S_j \ominus S'_j[z/u]$, since $z \not\ominus y$; but (–10a) was a valid UG, so y is internal. We already know that x is internal, so (–11) is a valid UG. Morover both of the affected variables, x and y are different from z.

Case 2.2, z is the same as x and therefore different from y.

In this case (–10a) becomes

$$S_i \ominus (\forall y)(S_j[z/y]) \tag{–12a}$$

and (–10b)

$$\begin{array}{lll} S'_i \ominus ((\forall y)(S_j[z/y]))[z/u] & & \\ \ominus (\forall y)(S_j[z/y][z/u]) & \text{since } z \not\ominus y & \\ \ominus (\forall y)(S_j[z/y]) & \text{since } z \text{ not free in } S_j[z/y] & \\ \ominus (\forall y)(S'_j[u/z][z/y]) & \text{from (–8)} & \\ \ominus (\forall y)S'_j[u/y] & \text{since } z \text{ not free in } S'_j & \text{(–12b)} \end{array}$$

Now (–12a) is a valid UG, so y is internal. Also u is fresh, so (–12b) is a valid UG. Moreover both affected variables, u and y, are different from z.

Case 2.3, z is the same as y and therefore different from x.

In this case (–10a) becomes

$$S_i \ominus (\forall y)(S_j[x/y]) \tag{–13a}$$

and (–10b)

$$\begin{array}{lll} S'_i \ominus ((\forall z)(S_j[x/z])[z/u] & & \\ \ominus (\forall z)(S_j[x/z]) & \text{since } z \text{ not free in this} & \\ \ominus (\forall z)(S'_j[u/z][x/z]) & \text{as usual.} & \text{(–13b)} \end{array}$$

It takes several steps to justify this. Here x is internal and u is fresh, so we may apply UG to S'_j to get $(\forall x)S'_j[u/x]$ and the PL6 to get $S'_j[u/x]$. And now we apply UG again to this to get $(\forall z)(S'_j[u/x][x/z])$ and $S'_j[u/x][x/z]$ is the same as $S'_j[u/z][x/z]$. Here x is internal and u is fresh, so we may apply UG to S'_j to get $(\forall x)S'_j[u/x]$; note that both affected variables u and x are different from z.

Next apply PL6 to get $S'_j[u/x]$.

Finally we apply UG again to this to get $(\forall z)(S'_j[u/x][x/z])$. We note that z is not free in S'_j, since it is $S_j[z/u]$, and so it is not free in $S'_j[u/x]$ either. So the only affected variable in this last UG is x and that is both internal and different to z. But $S'_j[u/x][x/z]$ is the same as $S'_j[u/z][x/z]$, which is (–13b).

(ii) Apply Part (i) m times. ■

C.4 Theorem

$$H_1, H_2, \ldots, H_h \vdash P \tag{–1}$$

(There is a proof of the deduction from $H_1, H_2, \ldots, H_h$ to P.)

if and only if

$$\frac{H_1, H_2, \ldots, H_h}{P} \qquad (-2)$$

(In *any* proof in which $H_1, H_2, \ldots, H_h$ appears, P may be asserted.)

Proof. That (–2) implies (–1) is trivial: to construct the required proof of (–1), simply write down all the hypotheses $H_1, H_2, \ldots, H_h$ followed by P, using (–2) as the justification of the last line.

Now to see that (–1) implies (–2).

> In Chapter 2 we argued thus: Suppose we have a proof of (–1), and a new proof in which $H_1, H_2, \ldots, H_h, P$ appear in that order, the appearance of P being justified by the rule (–2). Then the recipe for converting this into a proper proof was to simply replace the occurrence of P by the entire proof of (–1). This is a correct argument for Sentential Logic because the proof of (–1) remains valid in the context of the new proof. However in the case of Predicate Logic there is a possibility that the proof of (–1) might not remain valid in the context of a larger proof; specifically, it might contain an instance of UG in which the variable being generalised does not appear in any of the hypotheses $H_1, H_2, \ldots, H_h$ but does appear in one of the hypotheses of the larger proof. Indeed, this problem may occur several times in the same proof.
> The following argument avoids this problem.

Suppose we have a deduction,

$$K_1, K_2, \ldots, K_k \vdash C \qquad (-3)$$

with what we hope is a proof,

$$T_1, T_2, \ldots, T_t \qquad \text{so that } T_t \ni C \qquad (-4)$$

in which every step is a valid proof-step of (–3) except that some of them may be justified new Rule (–2) above. We show how to replace each such step by a sequence of valid proof steps (i.e. which do not use the new rule).

So, let T_p be a step in this proof which is justified by the new rule. That means that we already have a proof

$$S_1, S_2, \ldots, S_n \qquad (-5)$$

of a deduction

$$H_1, H_2, \ldots, H_h \vdash T_p \qquad (-6)$$

(so that $S_n \ni T_p$) and all the hypotheses $H_1, H_2, \ldots, H_h$ are among the preceding proof steps $T_1, T_2, \ldots, T_{p-1}$. We show how to replace T_p by a sequence of valid steps.

(We would like to simply replace T_p by the whole sequence (–5), but then we could get some trouble with variable symbols. Specifically, the proof (–5) might contain a UG step in which the variable symbol being generalised occurs free in the hypotheses $K_1, K_2, \ldots, K_k$. We can fix this by changing these variable symbols first.)

Let $x_1, x_2, \ldots, x_m$ be all the internal variable symbols in the proof (–5) and let $u_1, u_2, \ldots, u_m$ be a list of the same number of fresh variable symbols, in the sense that none of the u_i occur anywhere, free or bound, in anything we have seen so far, that is, in any of the

hypotheses $K_1, K_2, \ldots, K_k$, in any of the proof steps $T_1, T_2, \ldots, T_p$, in any of the hypotheses $H_1, H_2, \ldots, H_h$ or any of the proof steps $S_1, S_2, \ldots, S_n$.

Now, we substitute the new variable symbols $u_1, u_2, \ldots, u_m$ for the internal ones $x_1, x_2, \ldots, x_m$ throughout the proof (–5). to get a new sequence

$$S'_1, S'_2, \ldots, S'_n$$

and place it end-to-end with the sequence (–5) thus:

$$S'_1, S'_2, \ldots, S'_n, S_1, S_2, \ldots, S_n\,; \tag{–7}$$

Lemma C.3 tells us that this is a valid proof of of $H_1, H_2, \ldots, H_n \vdash S_n$ in which none of the variable symbols $x_1, x_2, \ldots, x_m$ appear as the variable being generalised in any application of UG.

We now substitute this sequence (–7) followed by sequence (–5) for T_p in the proof (–4):

$$T_1, T_2, \ldots, T_{p-1}, S'_1, S'_2, \ldots, S'_n, S_1, S_2, \ldots, S_n, T_{p+1}, \ldots, T_t\,. \tag{–8}$$

Note that T_p is still there, since it is the same as S_n.

It is now easy to check that all the new steps $S'_1, S'_2, \ldots, S'_n, S_1, S_2, \ldots, S_n$ here are valid proof-steps in a proof of (–3).

Any one of these steps which is an instance of an axiom in (–5) is still an instance of that axiom in (–8).

Any step which was a hypothesis in (–5)is the same as one of $H_1, H_2, \ldots, H_h$, and these are among $T_1, T_2, \ldots, T_{p-1}$, so this step is now justified by repetition.

Any step which followed by Modus Ponens in (–5) still does in (–8).

Finally, in any step which follows by UG in (–5), the affected variables are among the fresh variable symbols u_i, and these have been chosen so that none of them occur free in any of $K_1, K_2, \ldots, K_k$, so the UG is still valid. ■

C.5 Corollary

If

$$H_1, H_2, \ldots, H_k \vdash P$$

then

$$H_1, H_2, \ldots, H_k, H_{k+1} \vdash P\ .$$

This says that, if we have a deduction of an expression P from hypotheses, we can always add an extra hypothesis and still have the deduction. As remarked above, the proof connecting them may have to be changed. Note that, by using this result repeatedly, we see that we can add as many redundant hypotheses as we like.

D Some theorems and metatheorems

D.1 Theorem

Informal statement Any theorem or deduction of **SL** is a theorem or deduction (respectively) of any first order theory; that of course includes **PL**. Such proofs only use Axioms PL1, PL2 and PL3 and the rule MP.

Comment The wording of this theorem needs to be cleaned up a bit. For example, $p \Rightarrow p$ is a theorem of **SL**, and so is also a theorem of **PL** — provided the proposition symbol p appears in **PL** as a nullary relation. But our theorem is of course meant to include examples such as $x = y \;\Rightarrow\; x = y$ and $(\forall x)P(x) \Rightarrow (\forall x)P(x)$. These are not covered by the statement above because they contain symbols which are not part of the language **SL** and therefore cannot be theorems of **SL**.

Proper statement Any theorem or deduction *schema* of **SL** is also a theorem or deduction schema of **PL**. (In **PL** one substitutes expressions of **PL** for the expression-letters in the schema.) Such proofs only use Axioms PL1, PL2 and PL3 and the rule MP.

Proof. Inspection of the definitions of proofs in **SL** and first order languages shows that a proof in **SL** is a proof in any first order language, without need for any change. ■

In proofs and arguments henceforth, it will be assumed that you are familiar with Sentential Logic. Steps in proofs which depend on sentential logic arguments alone are simply notated "by **SL**" or something similar.

D.2 Theorem: Substitution of dummies

$$(\forall x)P \;\vdash\; (\forall y)P[x/y]$$

provided the substitution $[x/y]$ is acceptable in P and that either $y \ominus x$ or y does not occur free in P.

(In the alternative notation this is $(\forall x)P(x) \;\vdash\; (\forall y)P(y)$.)

Proof. Note that saying that either $y \ominus x$ or y does not occur free in P is the same as saying that y does not occur free in $(\forall x)P$.

1	$(\forall x)P$	hyp
2	P	PL6 and MP
3	$(\forall y)P[x/y]$	UG

■

D.3 Theorem: Existential Generalisation

Suppose that the substitution $[x/t]$ is acceptable in P (where t is a term). Then

$$P[x/t] \;\vdash\; (\exists x)P\,.$$

Proof. It is enough to show that $\vdash P[x/t] \Rightarrow (\exists x)P$. Using the definition of $\exists$ (in Section A.3), this is $\vdash P[x/t] \Rightarrow \neg(\forall x)\neg P$.

▶ A.3

Now

1	$(\forall x)\neg P \Rightarrow \neg P[x/t]$	PL6
2	$P[x/t] \Rightarrow \neg(\forall x)\neg P$	SL

1. $(\forall x)\neg P \Rightarrow \neg P[x/t]$ PL6
2. $P[x/t] \Rightarrow \neg(\forall x)\neg P$ SL. ■

D.4 Theorem

$\vdash \quad (\forall x)(P \Leftrightarrow Q) \;\Rightarrow\; ((\forall x)P \Leftrightarrow (\forall x)Q)$

Proof. First we prove $(\forall x)(P \Leftrightarrow Q) \vdash ((\forall x)P \Rightarrow (\forall x)Q)$.

1	$(\forall x)(P \Leftrightarrow Q)$	hyp
2	$P \Leftrightarrow Q$	PL6 on 1
3	$(\forall x)P$	subhyp
4	P	PL6 on 3
5	Q	MP on 4 and 2
6	$(\forall x)Q$	UG
7	$(\forall x)P \Rightarrow (\forall x)Q$	Ded: 3–6

By a similar argument

$$(\forall x)(P \Leftrightarrow Q) \quad \vdash \quad (\forall x)Q \Rightarrow (\forall x)P\,.$$

Now by **SL**

$$(\forall x)(P \Leftrightarrow Q) \quad \vdash \quad (\forall x)Q \Leftrightarrow (\forall x)P\,.$$

The result follows by an application of the Deduction Theorem. ■

D.5 Theorem: Substitution of Equivalents

Suppose that P, P' and Q are expressions such that P occurs as a subexpression of Q and Q' is the expression which results from substituting P' for P in Q. Suppose further that this substitution is "acceptable" in the sense that, if P lies within the scope of a quantifier $(\forall x)$ in Q, then neither P nor P' contain a free occurrence of x. Then

$$P \Leftrightarrow P' \quad \vdash \quad Q \Leftrightarrow Q'\,.$$

Proof. This is by induction over the way Q is built up from P.

Suppose first that $Q \ominus P$. Then $Q' \ominus P'$ and the result is trivially true.

Now suppose that Q is $\neg R$, where P is a subexpression of R. Then Q' is $\neg R'$, where R' is the result of substituting P' for P in R. This substitution is acceptable in R, so inductively $R \Leftrightarrow R'$ and $Q \Leftrightarrow Q'$ follows by **SL**.

Now suppose that Q is $\neg P$. Then Q' is $\neg P'$ and we need to prove that $P \Leftrightarrow P' \vdash \neg P \Leftrightarrow \neg P'$; this is easy **SL**.

If Q is $R \Rightarrow S$, where P is a subexpression of either R or S, we may use a similar argument.

Finally, if Q is $(\forall x)R$, where P is a subexpression of R, then the acceptability condition means that x does not occur free in either P or P'. By the inductive hypothesis we have $R \Leftrightarrow R'$. Now we prove that

$$R \Leftrightarrow R' \quad \vdash \quad Q \Rightarrow Q' \ .$$

that is

$$R \Leftrightarrow R' \quad \vdash \quad (\forall x)R \Rightarrow (\forall x)R' \ .$$

1	$R \Leftrightarrow R'$	hyp
2	$(\forall x)R$	subhyp
3	R	PL6 on 2
4	R'	MP: 3 and 1
5	$(\forall x)R'$	PL5 on 4
6	$(\forall x)R \Rightarrow (\forall x)R'$	Ded: 2–5

By the same argument, $R \Leftrightarrow R' \vdash (\forall x)R' \Rightarrow (\forall x)R$.

From these, by **SL**, $R \Leftrightarrow R' \vdash (\forall x)R \Leftrightarrow (\forall x)R'$, the required result. ■

D.6 Theorem

(i) $\vdash (\forall x)P \Leftrightarrow \neg(\exists x)\neg P$

(ii) $\vdash (\forall x)\neg P \Leftrightarrow \neg(\exists x)P$

(iii) $\vdash \neg(\forall x)P \Leftrightarrow (\exists x)\neg P$

Proof. These all follow from the definition of $\exists$ by **SL**. ■

E Quantifying non-free variable symbols

The theorems in this section are special. They deal with the cases in which an expression occurs inside a quantifier, but does not mention the quantified variable. A simple example is an expression such as

$$(\forall x)(\,2+2=4\,)\,. \tag{–1A}$$

Such expressions turn up from time to time (often while simplifying more complicated expressions), so we need to be able to deal with them. Luckily, that is usually quite simple. Basically, a statement such as the one above is equivalent to the same one without the quantifier, that is

$$2+2=4\,. \tag{–1B}$$

More generally, these remarks apply to expressions in which the variable in question does not occur free. For example, in

$$(\forall x)(\forall y)(\,x+y=y+x\,)\,. \tag{–2A}$$

neither x nor y is free. Consequently this expression is equivalent to

$$(\forall x)(\forall x)(\forall y)(\,x+y=y+x\,)\,. \tag{–2B}$$

and to

$$(\forall y)(\forall x)(\forall y)(\,x+y=y+x\,)\,. \tag{–2C}$$

This is what Theorem E.1 below tells us.

More generally still, similar simplifications occur when a quantifier is applied to an expression part of which does not mention the quantified variable: usually that part of the expression can simply be "taken outside" of the scope of the quantifier. The other theorems in this section are of this nature.

Warning *In the following theorems, the expression called S does not contain the variable x free; the other expression P might contain x free. These theorems* ***do not hold*** *if S contains x free.*

E.1 Theorem

Suppose that the variable x does not occur free in S. Then

(i) $\vdash\ S \Leftrightarrow (\forall x)S$

(ii) $\vdash\ S \Leftrightarrow (\exists x)S$.

Proof. (i) $\vdash S \Rightarrow (\forall x)S$ by PL5. Conversely $\vdash (\forall x)S \Rightarrow S$ by PL6. The result follows.

(ii) Applying (i) to $\neg S$ we have $\vdash \neg S \Leftrightarrow (\forall x)S$. But $\vdash (\forall x)\neg S \Leftrightarrow \neg(\exists x)S$ and so we have $\vdash \neg S \Leftrightarrow \neg(\exists x)S$. The result now follows by **SL**. ■

E.2 Theorem

Suppose that the variable x does not occur free in S (but may occur free in P). Then

(i) $\vdash (\forall x)(S \wedge P) \Leftrightarrow S \wedge (\forall x)P$

(ii) $\vdash (\forall x)(S \vee P) \Leftrightarrow S \vee (\forall x)P$

(iii) $\vdash (\exists x)(S \wedge P) \Leftrightarrow S \wedge (\exists x)P$

(iv) $\vdash (\exists x)(S \vee P) \Leftrightarrow S \vee (\exists x)P$

(This may look clearer in the alternative notation: suppose that the variable x does not occur free in S. Then

(i) $\vdash (\forall x)(S \wedge P(x)) \Leftrightarrow S \wedge (\forall x)P(x)$

(ii) $\vdash (\forall x)(S \vee P(x)) \Leftrightarrow S \vee (\forall x)P(x)$

(iii) $\vdash (\exists x)(S \wedge P(x)) \Leftrightarrow S \wedge (\exists x)P(x)$

(iv) $\vdash (\exists x)(S \vee P(x)) \Leftrightarrow S \vee (\exists x)P(x))$

Proof. **(i)** First we prove that $(\forall x)(S \wedge P) \vdash S \wedge (\forall x)P$.

1	$(\forall x)(S \wedge P)$	hyp
2	$S \wedge P$	PL6 on 1
3	S	SL on 2
4	P	SL on 2
5	$(\forall x)P$	UG on 4
6	$S \wedge (\forall x)P$	SL on 3 and 5

Next we prove that $S \wedge (\forall x)P \vdash (\forall x)(S \wedge P)$.

1	$S \wedge (\forall x)P$	hyp
2	S	SL on 1
3	$(\forall x)P$	SL on 1
4	P	PL6 on 3
5	$S \wedge P$	SL on 2 and 4
6	$(\forall x)(S \wedge P)$	UG on 5

The result now follows.

(ii) First we prove that $(\forall x)(S \vee P) \vdash S \vee (\forall x)P$ in the form $(\forall x)(\neg S \Rightarrow P) \vdash \neg S \Rightarrow (\forall x)P$

1	$(\forall x)(\neg S \Rightarrow P)$	hyp
2	$\neg S \Rightarrow P$	PL6 on 1
3	$\neg S$	subhyp
4	P	MP: 3 and 2
5	$(\forall x)P$	UG on 4
6	$\neg S \Rightarrow (\forall x)P$	Ded: 3–5

Now we prove $S \vee (\forall x)P \vdash (\forall x)(S \vee P)$.

1	$S \vee (\forall x)P$	hyp
2	S	subhyp
3	$S \vee P$	SL on 2
4	$(\forall x)(S \vee P)$	UG on 3
5	$S \Rightarrow (\forall x)(S \vee P)$	Ded: 2–4
6	$(\forall x)P$	subhyp
7	P	PL6 on 6
8	$S \vee P$	SL on 7
9	$(\forall x)(S \vee P)$	UG on 8
10	$(\forall x)P \Rightarrow (\forall x)(S \vee P)$	Ded: 6–9
11	$(\forall x)(S \vee P)$	Proof by Cases: 1,5 and 10

The result now follows.

(iii) and **(iv)** follow from **(i)** and **(ii)** by De Morgan's laws. ■

E.3 Theorem

Suppose that the variable x does not occur free in S (but may occur free in P). Then

(i) $\vdash (\forall x)(S \Rightarrow P) \Leftrightarrow (S \Rightarrow (\forall x)P)$

(ii) $\vdash (\forall x)(P \Rightarrow S) \Leftrightarrow ((\exists x)P \Rightarrow S)$

Proof. **(i)** Using the theorem of **SL** that $(A \Rightarrow B) \Leftrightarrow (\neg A \vee B)$, this is the same as

$$\vdash (\forall x)(\neg S \vee P) \Leftrightarrow \neg S \vee (\forall x)P$$

and this is a special case of Theorem D9(ii).

Abbreviated proof of (ii)

$$
\begin{array}{lll}
(\forall x)(P \Rightarrow S) \Leftrightarrow & (\forall x)(\neg S \Rightarrow \neg P) & \text{by SL} \\
\Leftrightarrow & \neg S \Rightarrow (\forall x)\neg P & \text{just proved} \\
\Leftrightarrow & \neg S \Rightarrow \neg(\exists x)P & \text{Theorem D7(ii)} \\
\Leftrightarrow & (\exists x)P \Rightarrow S & \text{by SL}
\end{array}
$$

■

Note that the forward direction of (ii),

$$\vdash \quad (\forall x)(P \Rightarrow S) \Rightarrow ((\exists x)P \Rightarrow S)$$

can be rewritten

$$(\forall x)(P \Rightarrow S)\ ,\ (\exists x)P \quad \vdash \quad S$$

which is a sort of quantified version of the Constructive Dilemma.

F Deductions involving choice

F.1 Example

Consider the (ordinary mathematical) proof of the result in group theory: Let A and B be subgroups of a group G such that $A \cup B$ is a subgroup also. Then either $A \subseteq B$ or $B \subseteq A$.

> *Proof* Suppose not, that is, that neither A nor B is a subset of the other. Then there are group members $a \in A$ and $b \in B$ such that $a \notin B$ and $b \notin A$. Since a and b are members of the subgroup $A \cup B$, we have $ab \in A \cup B$ and so either $ab \in A$ or $ab \in B$; suppose without loss of generality that $ab \in A$. Now remember that $a \in A$ also, so $b = a^{-1}.ab \in A$, a contradiction.

Comments There are a number of interesting aspects of this proof (proof by contradiction, use of the Constructive Dilemma, apparantly unavoidable use of Reductio ad Absurdem and so on). The part of interest now is the assertion of the existence of a and b. Making the assertion of the existence of a a little bit more formal, it goes

> $A \not\subseteq B$
> Therefore $\neg(\forall x)(x \in A \Rightarrow x \in B)$ from the definition of $B \subseteq A$
> and so $(\exists x)(x \in A \ \wedge \ x \notin B)$ by PL manipulation.
> Now let a be an element such that $a \in A$ and $a \notin B$.
> etc ...

This is a common form of argument in mathematics. In general, it is as follows. Suppose that a line $(\exists x)C(x)$ has been established in a proof. We then go on to say something like "let a be an object such that $C(a)$" and continue the proof, finally arriving at a conclusion that does not mention a. There is no rule discussed so far which (directly) justifies such a step in a proof. We now show that such a rule can be introduced, and how to do it. There are two things to do: first, decide exactly what the rule should be and, second, prove (by a metatheorem) that the rule is a genuine derived rule.

Observe first that our "object a" is treated like a temporary constant, not a variable — the formal difference between how constants and variables may be treated is simply that constants may not be quantified. If we are to introduce a new constant into our language on the fly, this involves extending the language (and then we immediately have to extend our sets of strings, expressions and axiom schema). This is a bit unwieldy, so instead we adopt the artifice of using a new variable symbol (that is, a variable symbol not used in this proof so far) and adding a condition to the new rule that such a variable must be treated like a constant, that is, it may not be quantified. Since we have a countably infinite number of variable symbols at our disposal, this does not involve us in extending the language.

In the course of a deduction, if a statement $(\exists x)C(x)$ has been asserted, we are now allowed to assert $C(a)$, where a is a new variable symbol (not used before in this proof) and the whole is subject to some good behaviour rules. We will say that the *Choice Rule* has been applied to $(\exists x)C(x)$ to yield $C(a)$ and that a is the *choice variable* involved.

F.2 Definition: Deduction involving choice

Our new definition of a deduction (involving choice) is as follows. We say that

$$H_1, H_2, \ldots, H_k \vdash A \qquad \text{(with choice)}$$

if there is a sequence $L_1, L_2, \ldots, L_n$ of expressions such that L_n is A and the following conditions hold:

(1) For each $i = 1, 2, \ldots, n$, one of the following holds

(i) L_i is an axiom of the theory,

(ii) L_i is one of the hypotheses $H_1, H_2, \ldots, H_k$, or

(iii) L_i follows by MP, UG or the Choice Rule from earlier expressions in the sequence.

(2) In any application of UG (to $P(x)$ to yield $(\forall x)P(x)$ say), the variable x being generalised must not occur free in any of the hypotheses $H_1, H_2, \ldots, H_k$. Moreover it must not occur free in any line of the proof of the form $(\exists x)C(x)$ to which the Choice Rule has been applied.

(3) In any application of the Choice Rule (to $(\exists x)C(x)$ to yield $C(a)$ say), the variable a must occur there for the first time in the proof, that is, it must not occur in any of the hypotheses, nor in any of the preceding lines in the proof.

(4) The statement A being proved must not contain any of the choice variables a which occur in the proof.

F.3 Notes

(i) The rules effectively make a choice variable behave like a new constant.

(ii) Under this definition, a deduction (with choice) in which the Choice Rule is not in fact used, is just an ordinary deduction. Therefore this new kind of deduction is an extension of the old kind.

F.4 Theorem

$$H_1, H_2, \ldots, H_k \vdash A \qquad \text{(with choice)} \qquad (-1)$$

if and only if

$$H_1, H_2, \ldots, H_k \vdash A \qquad \text{(without choice)} \qquad (-2)$$

Proof. It has just been observed that a deduction without choice is a special case of a deduction with it, so if (–2) then (–1). We now prove that if (–1) then (–2), by induction over the number of applications of the Choice Rule in the assumed proof with choice

$$B_1, B_2, \ldots, B_n$$

of (–1). If the Choice Rule is not used in the proof, then it is already a proof without choice and (–2) is the case. Now suppose that the Choice Rule is used at least once; suppose further that the first use of the rule in the proof is to $(\exists x)C(x)$ to yield $C(y)$. Now note that

$$H_1, H_2, \ldots, H_k, C(y) \vdash A \qquad \text{(with choice).} \qquad (-3)$$

This uses exactly the same proof, except that the line $C(y)$ is now justified by "Hyp" instead of " "choice rule". [*Not so fast.* We must check that $B_1, B_2, \ldots, B_n$ does indeed satisfy the conditions in the definition of a proof with choice of (–3). A careful reading of the conditions shows that this is in fact so.] Therefore the Choice Rule is used one less time in $B_1, B_2, \ldots, B_n$ as a proof of (–3) than it was as a proof of (–1). We may therefore use induction to conclude that

$$H_1, H_2, \ldots, H_k, C(y) \vdash A \qquad \text{(without choice).} \tag{–4}$$

We also note that $(\exists x)C(x)$ occurs in the proof before $C(y)$, and so that part of the proof from the beginning up to and including $(\exists x)C(x)$ constitutes a proof of that expression in which the Choice Rule is not used at all. In other words

$$H_1, H_2, \ldots, H_k \vdash (\exists x)C(x) \qquad \text{(without choice).} \tag{–5}$$

Now apply the (ordinary) Deduction Theorem to (–4)

$$H_1, H_2, \ldots, H_k \vdash C(y) \Rightarrow A$$

Now y occurs nowhere in $H_1, H_2, \ldots, H_k$, so by UG

$$H_1, H_2, \ldots, H_k \vdash (\forall y)(C(y) \Rightarrow A)$$

From Theorem 10(ii) and Substitution of Dummies,

$$H_1, H_2, \ldots, H_k \vdash (\exists x)C(x) \Rightarrow A$$

This, with (–5) and MP, gives (–2). ■

F.5 Corollary: The Deduction Theorem with Choice

If	$H_1, H_2, \ldots, H_k, A \vdash B$	(with choice)
then	$H_1, H_2, \ldots, H_k \vdash A \Rightarrow B$	(without choice)

Note What this means is that we can use the Choice Rule in deductions any time we like (as long as we observe the conditions laid down above). There is no longer any real need to preserve the distinction between a deduction with choice and one without it, so we will not do so.

F.6 Theorem

(i) $\vdash (\forall x)(\forall y)P \Leftrightarrow (\forall y)(\forall x)P$

(ii) $\vdash (\exists x)(\exists y)P \Leftrightarrow (\exists y)(\exists x)P$

(iii) $\vdash (\exists x)(\forall y)P \Rightarrow (\forall y)(\exists x)P$

Note The third statement here is an implication, not an equivalence. The reverse implication is not a theorem of **PL**.

Proof. **(i)** First we prove $(\forall x)(\forall y)P \vdash ((\forall y)(\forall x)P$.

1	$(\forall x)(\forall y)P)$	hyp
2	$(\forall y)P$	PL6 on 1
3	P	PL6 on 2
4	$(\forall x)P$	UG on 3
5	$(\forall y)(\forall x)P$	UG on 4

The proof of the converse is the same.

(ii) This follows from (i) by De Morgan's laws.

(iii) We prove $(\exists x)(\forall y)P \vdash (\forall y)(\exists x)P$.

In this proof, c is a variable symbol that does not appear in P.

1	$(\exists x)(\forall y)P$	Hyp
2	$(\forall y)P[x/c]$	Choice rule on 1
3	$P[x/c]$	PL6 on 2
4	$(\exists x)P$	EG on 3
5	$(\forall y)(\exists x)P$	UG (Note that x does not occur free in Line 1)

■

G Consistency

G.1 Theorem

Predicate Logic **PL** is consistent.

Proof. We show that Predicate Logic contains no expression P such that both $\vdash P$ and $\vdash \neg P$.

We define, for each expression P, a *value* $\|P\|$ which will be either 0 or 1. This is by induction over the construction of P.

- If P is atomic, $P \ominus r(t_1, t_2, \ldots, t_n)$ say, then $\|P\| = 1$.
- If $P \ominus \neg Q$ then $\|P\| = 1 - \|Q\|$.
- If $P \ominus Q \Rightarrow R$ then $\|P\| = \begin{cases} 1 & \text{if } \|Q\| \leq \|R\|; \\ 0 & \text{otherwise.} \end{cases}$
- If $P \ominus (\forall x)Q$ then $\|P\| = \|Q\|$. ■

G.2 Lemma

If the substitution $[x/t]$ is acceptable in P, then $\|P[x/t]\| = \|P\|$.

Proof. This follows the construction of P, using the "careful" description of acceptability given in A6.

If P is atomic, then so is $P[x/t]$ and so both have value 1.

If $P \ominus \neg Q$ and $[x/t]$ is acceptable in P, then it is acceptable in Q also and we have $\|P[x/t]\| = \|\neg Q[x/t]\| = \|\neg Q\| = \|P\|$.

Similarly, if $P \ominus (Q \Rightarrow R)$ and $[x/t]$ is acceptable in P, then it is acceptable in Q and R also and we have

$$\|P[x/t]\| \;=\; \|Q[x/t] \Rightarrow R[x/t]\| \;=\; 1 \qquad \text{if and only if} \qquad \|Q[x/t]\| \leq \|R[x/t]\|$$

and so, since $\|Q[x/t]\| = \|Q\|$ and $\|R[x/t]\|$, $\|P[x/t]\| = 1$ if and only if $\|Q\| \leq \|R\|$, that is, $\|P[x/t]\| = \|Q \Rightarrow R\| = \|P\|$.

If $P \ominus (\forall y)Q$, where $y \notin x$ and $[x/t]$ is acceptable in P, then it is acceptable in Q also and the variable symbol y does not occur in t. Then

$$\|P[x/t]\| \;=\; \|(\forall y)Q[x/t]\| \;=\; \|Q[x/t]\| \;=\; \|Q\| \;=\; \|(\forall y)Q\| \;=\; \|P\|\,.$$

Finally, if $P \ominus (\forall x)Q$ then $[x/t]$ is acceptable in P, $\|P[x/t]\| = \|P\|$ and the result is trivially true. ■

G.3 Lemma

The value of every theorem is 1.

Proof. This is by induction over the length of the proof of the theorem. It is enough to prove that the value of every axiom is 1 and that, if the expression P follows by Modus Ponens or Universal Generalisation from expressions of value 1, then $\|P\| = 1$ also.

The proofs for Axioms PL1, PL2 and PL3 are simply by checking the various cases for values of $\|P\|$, $\|Q\|$ and $\|R\|$.

[Axiom PL4: $(\forall x)(P \Rightarrow Q) \ \Rightarrow \ ((\forall x)P \Rightarrow (\forall x)Q)$].
Since $\|(\forall x)(P \Rightarrow Q)\| = \|P \Rightarrow Q\| = 1$ if and only if $\|P\| \leq \|Q\|$ and $\|(\forall x)P \Rightarrow (\forall x)Q\| = 1$ if and only if $\|(\forall x)P\| \leq \|(\forall x)Q\|$, that is, if and only if $\|P\| \leq \|Q\|$, we have $\|(\forall x)(P \Rightarrow Q)\| = \|(\forall x)P \Rightarrow (\forall x)Q\|$ and so $\|(\forall x)(P \Rightarrow Q) \ \Rightarrow \ ((\forall x)P \Rightarrow (\forall x)Q)\|$, as required.

[Axiom PL5: $P \Rightarrow (\forall x)P$, where x does not occur free in P].
Then $\|P\| = \|(\forall x)P\|$ and the result is trivial.

[Axiom PL6: $(\forall x)P \Rightarrow P[x/t]$, where $[x/t]$ is acceptable in P].
Then $\|(\forall x)P\| = \|P\| = \|P[x/t]\|$ by Lemma 1, and again the result is trivial.

[Modus Ponens: There are expressions Q and $Q \Rightarrow P$, both of value 1].
Since $\|Q \Rightarrow P\| = 1$, we have $\|Q\| \leq \|P\|$. But also $\|Q\| = 1$, so $\|P\| = 1$.

[Universal Generalisation: $P = (\forall y)Q[x/y]$, where $\|Q\| = 1$, $[x/y]$ is acceptable in Q and, if $y \notin x$, y is not free in Q].
Then $\|P\| = \|(\forall y)Q[x/y]\| = \|Q[x/y]\| = \|Q\|$ (by Lemma 1). ■

Proof of the theorem. If P is a Theorem then $\|P\| = 1$, therefore $\|\neg P\| = 0$ and therefore $\neg P$ is not a theorem. ■

H Extensions

H.1 Definition

An *extension* of a deductive system **A** is a deductive system **B** with the same underlying language as **A** (that is, the same symbols and expressions) and such that every theorem of **A** is a theorem of **B**.

H.2 Lemma

Suppose that **A** is a consistent theory and that B is a fully quantified expression such that $\neg B$ is not provable in **A**. Then the theory which results from adding B as an extra axiom to **A** is also consistent.

Proof. Suppose that the new theory is not consistent. Then there is an expression, P say, such that $P \wedge \neg P$ is provable in the new theory. But then $B \vdash P \wedge \neg P$ in **A**. But from this it follows that $\neg B$ is provable in **A**. ■

H.3 Theorem

If **A** is a consistent first order theory, then it has a consistent complete extension.

Note that the extension so formed is not guaranteed to have a decidable set of axioms.

Proof. We suppose that we have formed an enumeration of the fully quantified expressions (that is, expressions with no free variables) of **A**, $A_1, A_2, \ldots$ say. Now define a sequence $\mathbf{B}_0, \mathbf{B}_1, \mathbf{B}_2, \ldots$ of deductive systems as follows. First $\mathbf{B}_0 = \mathbf{A}$. Now assume that $\mathbf{B}_n$ has been defined and we define $\mathbf{B}_{n+1}$. Consider the expression A_{n+1}; if $\neg A_{n+1}$ is not a theorem of $\mathbf{B}_n$, construct $\mathbf{B}_{n+1}$ by adding A_{n+1} as an extra axiom to $\mathbf{B}_n$ and if $\neg A_{n+1}$ is a theorem of $\mathbf{B}_n$, set $\mathbf{B}_{n+1} = \mathbf{B}_n$. Proceeding in this way, every $\mathbf{B}_i$ is a first order theory and, for $i < j$, $\mathbf{B}_j$ is an extension of $\mathbf{B}_i$. If we write $\mathbf{B}^*$ for the theory obtained by taking all the axioms of all the $\mathbf{B}_i$, then $\mathbf{B}^*$ is an extension of all the $\mathbf{B}_i$ including $\mathbf{B}_0 = \mathbf{A}$. We show that $\mathbf{B}^*$ is consistent and complete.

To see that $\mathbf{B}^*$ is consistent, it is enough to show that each $\mathbf{B}_n$ is consistent, since a proof of inconsistency involves only a finite number of axioms — and this follows from the lemma above.

To see that $\mathbf{B}^*$ is complete, let P be any fully quantified expression. Then $P = A_n$ for some n. If $\neg A_n$ is provable in $\mathbf{B}_{n-1}$ then it is provable in $\mathbf{B}^*$. Otherwise $\neg A_n$ is not provable in $\mathbf{B}_{n-1}$ and then A_n is an axiom of $\mathbf{B}_n$ and so is provable in $\mathbf{B}^*$. ■

I Comments on some of the notation

I.1 Binding

▶ A.9

Binding a variable with a quantifier turns it into a "dummy" variable. This is by no means peculiar to formal logic: For familiar examples, $\int \dots dx$ binds the variable x and $\sum_{i=1}^{n}$ binds i.

Here is an elementary property of integration:

$$\int (f(x) + g(x))dx \;=\; \int f(x)\,dx + \int g(x)\,dx \;. \tag{–1A}$$

The variable symbol x in fact has three different scopes, which can be thought of informally as different "meanings", here. The equation could equally well be written

$$\int (f(x) + g(x))dx \;=\; \int f(y)\,dy + \int g(z)\,dz \;. \tag{–1B}$$

At first glance, the second form (–1B) looks better, at least for a formal system, in that it does not require recognition of the various different scopes of the same variable symbol. On the other hand, the first form (–1A) is more in line with common usage. Furthermore, recognising different scopes is algorithmically quite simple (even if it allows us to write rather human-unfriendly expressions — see below) and, more importantly, legislating against expressions such as (–1A) in our formal language would require extra awkward little rules to be added. Therefore multiple different bindings of the same variable symbol *are* allowed in the formal language from the word go.

Moreover, if you really used a different letter for every different scope, you would probably run out of letters by the end of the first page.

For an example in formal logic, consider the deduction

$$P \;,\; Q \;\vdash\; P \,\&\, Q$$

By substitution, this allows

$$(\forall x)P \;,\; (\forall x)Q \;\vdash\; (\forall x)P \,\&\, (\forall x)Q$$

If multiple scopes were to be disallowed, this would have to become

$$(\forall x)P \;,\; (\forall x)Q \;\vdash\; (\forall x)P \,\&\, (\forall y)Q[x/y]$$

with various pernickity provisos on the choice of the (new?) variable symbol y.

We also allow variable symbols to occur both free and bound in the same expression, as in

$$P(x) \;\vee\; (\forall x)Q(x) \;. \tag{–2A}$$

This is not so common in ordinary mathematical notation, and could be considered to be (sometimes) confusing to humans. Compare

$$g(x) \;=\; x^2 + \int_0^1 f(x)\,dx \;. \tag{–3A}$$

Normal practice would suggest we write these

$$P(x) \ \vee \ (\forall y)Q(y) \ , \tag{–2B}$$

$$g(x) \ = \ x^2 + \int_0^1 f(t)\,dt \ , \tag{–3B}$$

however we do allow ourselves this freedom in the formal language, again because legislating against it causes more problems than it solves.

Note well that one is allowed to take expressions such as (–2A) and (–3A) and quantify them to get, for instance,

$$(\exists x)\big(P(x) \ \vee \ (\forall x)Q(x)\big) \ , \tag{–4A}$$

$$g(x) \ = \ \int_0^1 \left(x^2 + \int_0^1 f(x)\,dx\right) dx \ , \tag{–5A}$$

or even, I suppose,

$$h(x) \ = \ \int_0^x \int_0^x x^2 \,dx\,dx \ .$$

These expressions are perfectly OK — the scope of the "inner" x is unambiguously defined in each case.

The possible occurrence of the same variable symbol quantified in different ways in the same expression accounts for the care taken in the discussion of *free* and *bound* occurrences in Section A5 and what follows.

► A.11

I.2 Acceptability

Consider the equation

$$f(x) \ = \ \int_0^1 x \, \sin y \ dy \ .$$

We may substitute z for x here to get

$$f(z) \ = \ \int_0^1 z \, \sin y \ dy \ ,$$

but substituting y for x is not acceptable:

$$f(y) \ = \ \int_0^1 y \, \sin y \ dy$$

is a radically different statement (the right hand side is now a constant!).

In the last case, we are trying the substitution [x/y], when x occurs within the scope of the binding of y by the integral.

The same sort of thing occurs in logic. Suppose we are talking about the integers $\mathbb{Z}$. Here is an expression:

$$(\exists y)(x + y = 1) \ . \tag{6}$$

This is a statement about the number x, which is in fact true of all x (put $y = 1 - x$), so we could write

$$(\forall x)(\exists y)(x + y = 1) \ . \tag{7}$$

Suppose we substitute z for x in (6):

$$(\exists y)(z + y = 1) \ .$$

This says the same thing about z as (6) does about x and we have

$$(\forall z)(\exists y)(z + y = 1) \ .$$

All this is OK, however if we make the unacceptable substitution $[x/y]$ in (6) we get

$$(\exists y)(y + y = 1) \ .$$

which means something completely different — it has no free variable and is false.

So far we have looked at substitutions of one variable for another. We are also allowed to substitute terms for variables. Here are the results of the substitutions $[x/3]$ and $[x/x^2 + 3]$ in (6):

$$(\exists y)(3 + y = 1)$$
$$(\exists y)(x^2 + 3 + y = 1) \ .$$

These are OK, but the unacceptable substitution $[x/y^2 + 3]$, yielding

$$(\exists y)(y^2 + 3 + y = 1) \ .$$

is not.

In short, an expression with a free variable x can be thought of as saying something about that variable. An acceptable substitution $[x/t]$ yields an expression which says the same thing about the term t; an unacceptable substitution yields an expression which says something entirely different.

I.3 Axiom PL6

▶ A.13

This is the usual argument from general to particular. Suppose once more that we are talking about the integers $\mathbb{Z}$. The expression

$$(\forall x)(\exists y)(x + y = 0)$$

states that every number x has a negative ($y = -x$). We can validly deduce from this that 4 has a negative (by the substitution $[x/4]$)

$$(\exists y)(4 + y = 0)$$

or that $z + 1$ has a negative

$$(\exists y)(z + 1 + y = 0)$$

but the substitution $[x/y + 1]$ is not acceptable — using this PL6 would yield

$$(\exists y)(y + 1 + y = 0)$$

which does not say that $y + 1$ has a negative. (To say that $y + 1$ has a negative, we would use Theorems D2 and D5 to get $(\forall x)(\exists z)(x + z = 0)$, then make the acceptable use of PL6 on this to get $(\exists z)(y + 1 + z = 0)$.)

► A.13

I.4 Universal Generalisation

Suppose we are doing elementary set theory and have defined $A \subseteq B$ to mean $(\forall x)(x \in A \Rightarrow x \in B)$. Now we want to prove that $A \subseteq B$, $B \subseteq C \vdash A \subseteq C$. A typical argument would be:

> Let $x \in A$. Then the hypothesis $A \subseteq B$ gives $x \in A \Rightarrow x \in B$, and so $x \in B$. Now, in the same way, the hypothesis $B \subseteq C$ gives $x \in C$. We have proved that $x \in A \Rightarrow x \in C$ and this is true for any x, so $A \subseteq C$.

The phrase "let $x \in A$" here signals two things (in our informal language). Firstly the word "let" signals that we are about to start a subsidiary deduction, of the form $x \in A \vdash$ *something*, in this case $x \in A \vdash x \in C$, in order to deduce $x \in A \Rightarrow x \in C$ by the Deduction Theorem. Secondly the appearance of the new variable symbol x signals that we are going to be using UG on this variable. Once $x \in A \Rightarrow x \in C$ has been proved, the phrase "this is true for any x" is a second signal that UG is about to be used to deduce $(\forall x)(x \in A \Rightarrow x \in C)$.

What about the proviso that x must not occur free in any of the hypotheses? Consider the following, which can be proved in Number Theory:

$$x = 3 \quad \vdash \quad x \text{ is a prime number.}$$

So there is a proof leading from the hypothesis $x = 3$ to the conclusion x is a prime number. If we ignore the proviso, UG would allow us to add extra line $(\forall x)(x$ is a prime number) to this proof, yielding a proof of the deduction that, if $x = 3$, then every number is a prime number (which is absurd!).

4. SOME FIRST ORDER THEORIES

A First order theories with equality

A.1 Definition

A first order theory with equality is a first order theory with an extra binary relation $=$ and the extra axioms:

(Eq1) $(\forall x)(\ x = x\ )$

(Eq2) $(\forall x)(\forall y)(\forall z)(\ x = y\ \Rightarrow\ (P[z/x] \Rightarrow P[z/y])\ )$ provided both substitutions are acceptable.

Here "acceptable" means, as usual, that the occurrence of z being replaced is not within the scope of a $(\forall y)$ or a $(\forall z)$ quantifier in P.

Eq2 is an axiom schema: one axiom for every choice of expression P. On the other hand, Eq1 is a plain axiom.

As usual with axioms, we can use either the fully quantified version, as I have done above, or the unquantified version:

(Eq1u) $x = x$

(Eq2u) $x = y\ \Rightarrow\ (P[z/x] \Rightarrow P[z/y])$ provided both substitutions are acceptable.

(You can deduce the unquantified versions from the fully quantified ones by using PL6, and go in the opposite direction by UG.)

A.2 Partial substitution

Eq2 (or Eq2u if you prefer) above tells us that, if we know that $x = y$, then we can substitute y for x in any expression; if P is a theorem and we substitute y for x in P, that will be a theorem too. So why don't we use the simpler axiom

(Eq2??) $x = y\ \Rightarrow\ (P \Rightarrow P[x/y])$ provided that this substitution is acceptable ?

The reason is that the kind of substitution we are doing here is different from the kind we did before. In a nutshell, the substitution $[x/y]$ we have been using so far is a direction to replace *every* occurrence of x by y. Whereas, if we know that $x = y$, we only need replace some of them. A couple of examples may make this clear.

Firstly, an example of the application of PL6. Suppose we have proved that

$$(\forall x)(x + 1 > x)$$

From this we can deduce (by the substitution [x/y]) that

$$y + 1 > y$$

but we *must* make the substitution everywhere: we may not deduce (from PL6) that

$$x + 1 > y \ .$$

On the other hand, suppose we have proved that

$$x^2 - 1 - x > \sin x + e^{-x} \ ; \tag{1}$$

if we should also know that $x = y$, it is perfectly correct to deduce that

$$x^2 - 1 - y > \sin x + e^{-y} \tag{2}$$

(note that we have replaced some, but not all, of the x's) and we occasionally need to do this sort of thing. Here we are using Eq2 instead of PL6.

Now, our $[x/y]$ notation means replace *all* the x's by y's. Must we introduce another new notation for replacing only some of them? We can avoid this annoyance by a trick: we can use the old $[x/y]$ notation in a smart way. Suppose we have proved the expression (1) above and also $x = y$; we want to prove (2). Make a new expression (let's call it Q) from (1) by choosing a variable symbol that does not occur anywhere else in (1), z say, and replacing just the x's we want to change to y's by z's; that is

$$Q \quad \ominus \quad x^2 - 1 - z = \sin x + e^{-z} \ .$$

(I am not saying that this is a theorem, it is just an expression to play with.) But now (Eq2u) gives us

$$(x = y) \ \Rightarrow \ \big(Q[z/x] \Rightarrow Q[z/y]\big) \tag{3}$$

which is

$$(x = y) \ \Rightarrow \ \big((x^2 - 1 - x = \sin x + e^{-x}) \ \Rightarrow \ (x^2 - 1 - y = \sin x + e^{-y})\big) \ .$$

This is just what we need: we have proved $x = y$ and $x^2 - 1 - x = \sin x + e^{-x}$, so from the above $x^2 - 1 - y = \sin x + e^{-y}$ follows.

Putting this another way, (Eq2) gives us (3), and here we have already proved $x = y$ and $Q[z/x]$, so $Q[z/y]$ follows, and that is the expression we want to prove.

We can sum this discussion up by saying that (Eq2) can be stated another way:

(Eq2′) $(\forall x)(\forall y)(\ x = y \ \Rightarrow \ (P \Rightarrow Q)\ )$ where P is any expression in which x occurs free and Q is the result of replacing one or more, but not necessarily all, of the free occurrences of x in P by y, provided that this replacement is acceptable.

Here "acceptable" means as usual that those occurrences of x that are being replaced are not within the scope of a $\forall y$ in P.

A.3 Immediate corollaries

Note There are a number of immediate corollaries of this.

(Eq1′) $t = t$ for every term t.

And another corollary of Eq2 is

(Eq2″) $x = y \;\Rightarrow\; (P[z/x] \Leftrightarrow P[z/y])$ provided that both substitutions are acceptable.

(Eq3) $x = y \;\Rightarrow\; s[z/x] = s[z/y]$ where s is any term.

Again, this may be rephrased

(Eq3') $x = y \;\Rightarrow\; s = t$ where s is any term and t is the result of replacing one or more occurrences of x in s by y.

(Eq4) $x = y \;\Rightarrow\; y = x$.

(Eq5) $x = y \;\wedge\; y = z \;\Rightarrow\; x = z$.

Proofs (Eq1′) Easy: this follows from (Eq1) by PL6.

(Eq2″) Not so easy. (Eq2u) gives us

$$x = y \;\Rightarrow\; \big(P[z/x] \Rightarrow P[z/y]\big) ,$$

so it is enough to prove

$$x = y \;\Rightarrow\; \big(P[z/y] \Rightarrow P[z/x]\big) .$$

We introduce a completely new variable symbol w (that is, it does not occur anywhere in P and is different from x, y and z). Now define a new expression Q by

$$Q \quad \Ominus \quad P[z/w] \Rightarrow P[z/x] .$$

Now (Eq2), which is a schema, gives

$$x = y \;\Rightarrow\; \big(Q[w/x] \Rightarrow Q[w/y]\big) . \tag{1}$$

Observing that Theorem **2**.F.2(ii) can be written ▶ **2**.F.2

$$\big(\mathcal{A} \Rightarrow (\mathcal{B} \Rightarrow \mathcal{C})\big) \;\Rightarrow\; \big(\mathcal{B} \Rightarrow (\mathcal{A} \Rightarrow \mathcal{C})\big) ,$$

we have

$$Q[w/x] \;\Rightarrow\; \big(x = y \;\Rightarrow\; P[w/y]\big) .$$

But

$$Q[w/x] \quad \Ominus \quad P[z/x] \Rightarrow P[z/x] \tag{2}$$

which is a Theorem **2**.B.1, so now we have ▶ **2**.B.1

$$x = y \;\Rightarrow\; P[w/y] .$$

and

$$Q[w/y] \quad \Ominus \quad P[z/y] \Rightarrow P[z/x] , \tag{3}$$

so this is (Eq2″), as required.

In setting out this proof, I have glossed over some pesky details to do with the substitutions. Firstly, if Line (1) is to be a valid application of (Eq2), both the substitutions $Q[w/x]$ and $Q[w/y]$ must be acceptable. From the definition of Q,

$$Q[w/x] \quad \Subset \quad P[z/w][w/x] \Rightarrow P[z/x][w/x]$$

— double substitutions which we must be careful about! Consider $P[z/w][w/x]$ first. We are given that $P[z/x]$ is acceptable, so no free occurrence of z lies within the scope of an x quantifier. In $P[z/w]$, we replace those free occurrences of z with w and, since w was new, there are no other occurrences of w in $P[z/w]$. Since none of the free occurrences of z were in the scope of an x quantifier in P, clearly none of the occurrences of w in $P[z/w]$ are either. This substitution is therefore acceptable.

Now consider $P[z/x][w/x]$. This is easy. Since w is new, it does not occur at all in $P[z/x]$ and therefore the substitution $[w/x]$ in it is trivially acceptable.

Now let us look at one (2). What we know is that

$$Q[w/x] \quad \Subset \quad P[z/w][w/x] \Rightarrow P[z/x][w/x] \ .$$

We have just remarked that the substitution $P[z/w][w/x]$ consists of replacing every fee occurrence of z in P by w, then replacing that w by x. This is clearly the same as replacing every z in P by x in the first place. That is, $P[z/w][w/x] \Subset P[z/x]$, as required. And, again as remarked above, $P[z/x]$ does not contain any occurrences of w, so the substitution $[w/x]$ has no effect on it: $P[z/x][w/x] \Subset P[z/x]$. This verifies Line (2).

Line (3) is verified in the same way.

(Eq3) Let w be a new variable symbol, that is, one which does not occur anywhere in s and is different from both x and y. Define the expression P to be $s[z/x] = s[z/w]$. Now (Eq2) gives

$$x = y \ \Rightarrow \ \big(P[w/x] \Rightarrow P[w/y]\big)$$

and thus

$$P[w/x] \ \Rightarrow \ \big(x = y \ \Rightarrow \ P[w/y]\big) \ .$$

But $P[w/x]$ is $s[z/x] \Rightarrow s[z/x]$, which is a case of (Eq1′), so we have

$$x = y \ \Rightarrow \ P[w/y] \ ,$$

that is

$$x = y \ \Rightarrow \ \big(s[z/x] = s[z/y]\big) \ ,$$

as required. Since terms cannot contain quantifiers, all these substitutions are automatically acceptable.

(Eq4) and **(Eq5)** are proved in much the same way. For (Eq4), define P to be the expression $w = x$. For (Eq5), first notice that it is equivalent to

$$x = y \ \Rightarrow \ (y = z \ \Rightarrow \ x = z) \, .$$

Now define P to be the expression $w = z \ \Rightarrow \ x = z$. Note that quantifiers do not occur in these expressions, so the substitutions are automatically acceptable.

A.4 Definition: Unique existence

We can now formally define the idea expressed by "There is a unique x such that $P(x)$". Other ways of expressing the same idea are "There is exactly one x such that $P(x)$" or "There is one and only one x such that $P(x)$". This is commonly symbolised $(\exists!x)P(x)$and is defined to mean

$$(\exists x)P(x) \ \wedge \ (\forall x)(\forall y)\big(P(x) \wedge P(y) \ \Rightarrow \ x = y\big) \ .$$

(Here y is the first variable symbol which does not occur in $P(x)$.) Some writers use the notation $(\exists_1 x)P(x)$ or $(\exists 1x)P(x)$ for this idea.

It would be quite useful to split this up into two ideas:

$$\begin{array}{ll} (\exists x)P(x) & \text{there is at least one } x \text{ such that } P(x) \ . \\ (!x)P(x) & \text{there is at most one } x \text{ such that } P(x) \, , \end{array}$$

the latter being defined to mean $(\forall x)(\forall y)\big(P(x) \wedge P(y) \ \Rightarrow \ x = y\big)$.

Then $(\exists!x)P(x)$ means both, that is $(\exists x)P(x) \wedge (!x)P(x)$.

A.5 Definition: Definition by description

A common mathematical construction is as follows. Suppose we have an expression $P(u, x_1, x_2, \ldots, x_n)$ and we have proved that, for any $x_1, x_2, \ldots, x_n$, there is a unique u such that $P(u, x_1, x_2, \ldots, x_n)$. Then we can think of this as defining u as a function of $x_1, x_2, \ldots, x_n$: we introduce a new function symbol, f say, and write

$$u \ = \ f(x_1, x_2, \ldots, x_n) \qquad \text{to mean} \qquad P(u, x_1, x_2, \ldots, x_n) \ .$$

This procedure is valid, in a sense which is made precise in the next theorem. We will call it *defining the function f by description.*

A.6 Theorem: Definition by description is valid.

Let **A** be a theory with equality and suppose that $P(u, x_1, x_2, \ldots, x_n)$ is an expression in **A** with $n + 1$ free variables ($n \geq 0$) such that

$$\vdash \ (\forall x_1)(\forall x_2) \ldots (\forall x_n)(\exists!u)P(u, x_1, x_2, \ldots, x_n) \qquad \text{in } \mathbf{A} \, .$$

Let us construct a new theory **B** by

- Adding a new n-ary function symbol f to the symbols of the language **A** and extending the sets of terms and expressions accordingly, as dictated by Definitions **3**.A.4 and **3**.A.6, ▶ **3**.A.4 ▶ **3**.A.6
- Extending the axiom schemas PL1–PL6 and Eq2 and the rules Modus Ponens and Universal Generalisation to include all the new expressions so formed,
- Adding a new axiom

  $$(\forall x_1)(\forall x_2) \ldots (\forall x_n)(\forall u)\big(u = f(x_1, x_2, \ldots, x_n) \ \Rightarrow \ P(u, x_1, x_2, \ldots, x_n)\big) \ .$$

  Then the new theory **B** is equivalent to the old one **A** (in the sense of Definition 2.H1).

Proof. We observe that **B** is an extension of **A**, in the sense that the symbols of **A** are a subset of the symbols of **B** and the same goes for strings, terms, expressions, axioms, rules and therefore theorems.

Observe that the new axiom $(\forall u)\big(u = f(x_1, x_2, \ldots, x_n) \;\Rightarrow\; P(u, x_1, x_2, \ldots, x_n)\big)$ gives rise in the usual way to theorems

$\vdash_{\mathbf{B}}$ $(\forall u)\big(u = f(t_1, t_2, \ldots, t_n) \;\Rightarrow\; P(u, t_1, t_2, \ldots, t_n)\big)$ for any terms $t_1, t_2, \ldots, t_n$ of **B** .

Here "$\vdash_{\mathbf{B}}$" means "the following is a theorem of **B**". In the same way, the assumed theorem

$(\exists! u)P(u, x_1, x_2, \ldots, x_n)$ of **A** gives rise to theorems

$\vdash_{\mathbf{A}}$ $(\exists! u)P(u, t_1, t_2, \ldots, t_n)$ for any terms $t_1, t_2, \ldots, t_n$ of **A**, provided the substitutions $[x_1/t_1], [x_2/t_2] \ldots [x_n/t_n]$ are all acceptable in P .

Since all theorems of **A** are theorems of **B**, these are also theorems of **B**.

We now show that $\vdash P(f(t_1, t_2, \ldots, t_n), t_1, t_2, \ldots, t_n)$ in **B**, for all terms $t_1, t_2, \ldots, t_n$, provided that these substitutions are acceptable in P.

1.	$(\forall u)\big(u = f(t_1, t_2, \ldots, t_n) \;\Rightarrow\; P(u, t_1, t_2, \ldots, t_n)\big)$	New axiom
2.	$f(t_1, t_2, \ldots, t_n) = f(t_1, t_2, \ldots, t_n) \;\Rightarrow\; P(f(t_1, t_2, \ldots, t_n), t_1, t_2, \ldots, t_n)$	PL6 on 1
3.	$f(t_1, t_2, \ldots, t_n) = f(t_1, t_2, \ldots, t_n)$	Eq1′
4.	$P(f(t_1, t_2, \ldots, t_n), t_1, t_2, \ldots, t_n)$	MP: 3, 2

We now define the functions $\varphi : \mathbf{A} \to \mathbf{B}$ and $\psi : \mathbf{B} \to \mathbf{A}$ (required by Definition 2.H1). The definition of φ is easy: it is the identity, that is, to each expression A of **A**, $\varphi(A)$ is A.

Define a "simple f-term" to be a term in **B** of the form $f(t_1, t_2, \ldots, t_n)$, where the terms $t_1, t_2, \ldots, t_n$ do not mention f.

Let B be any atomic expression in **B**. We define $\psi(B)$ by induction over the number of occurrences of the symbol f in B. If f does not occur in B, then $\psi(B) \ominus B$. Otherwise B must contain at least one simple f-term; write B^* for the result of replacing the leftmost occurrence of a simple f-term, $f(t_1, t_2, \ldots, t_n)$ say, in B by the first variable u which does not occur in B or P. Then

$$\psi(B) \quad \ominus \quad \psi\big((\exists u)(P(u, t_1, t_2, \ldots, t_n) \;\wedge\; B^*)\big) \; .$$

We observe that $(\exists u)(P(u, t_1, t_2, \ldots, t_n) \;\wedge\; B^*)$ has one less occurrence of the symbol f than does B and so this inductive definition is good.

We now extend ψ to all expressions in **B** in the obvious way:—

	$\psi(\neg B)$	$\ominus$	$\neg\psi(B)$
	$\psi(B \Rightarrow B')$	$\ominus$	$\psi(B) \Rightarrow \psi(B')$
and	$\psi((\forall x)B)$	$\ominus$	$(\forall x)\psi(B)$.

We now show that, for any expression B in $\mathbf{B}$, we have $B \vdash\!\dashv \psi(B)$ in $\mathbf{B}$.

Consider atomic expressions first. If f does not occur in B, then $\psi(B)$ is B and the result is trivial. Otherwise let $f(t_1, t_2, \ldots, t_n)$ be the leftmost occurrence of a simple f-term in B, so that $\psi(B)$ is $\psi\big((\exists u)(P(u, t_1, t_2, \ldots, t_n) \wedge B^*)\big)$. Using induction, it is clearly sufficient to show that

$$B \quad \vdash\!\dashv \quad (\exists u)(P(u, t_1, t_2, \ldots, t_n) \wedge B^*) \qquad \text{in } \mathbf{B} \ .$$

Proof of $B \vdash (\exists u)(P(u, t_1, t_2, \ldots, t_n) \wedge B^*)$ in $\mathbf{B}$.

1.	$P(f(t_1, t_2, \ldots, t_n), t_1, t_2, \ldots, t_n)$	Proved above (note that the substitution is acceptable)
2.	B	Hyp
3.	$P(f(t_1, t_2, \ldots, t_n), t_1, t_2, \ldots, t_n) \wedge B$	SL, 1 and 2
4.	$(\exists u)(P(u, t_1, t_2, \ldots, t_n) \wedge B^*)$	ExGen

Proof of $(\exists u)(P(u, t_1, t_2, \ldots, t_n) \wedge B^*) \vdash B$ in $\mathbf{B}$.

1.	$P(u, t_1, t_2, \ldots, t_n) \wedge B^*$	Choice on Hyp
2.	$P(u, t_1, t_2, \ldots, t_n)$	SL on 1
3.	B^*	SL on 1
4.	$P(f(t_1, t_2, \ldots, t_n), t_1, t_2, \ldots, t_n)$	Proved above (note, acceptable)
5.	$(\exists!u)P(u, t_1, t_2, \ldots, t_n)$	The assumed theorem
6.	$(\forall x)(\forall y)(P(x, t_1, t_2, \ldots, t_n) \wedge P(y, t_1, t_2, \ldots, t_n) \ \Rightarrow \ x = y$	From 5 by SL
7.	$P(u, t_1, t_2, \ldots, t_n) \wedge P(f(t_1, t_2, \ldots, t_n), t_1, t_2, \ldots, t_n)$ $\Rightarrow \ u = f(t_1, t_2, \ldots, t_n)$	PL6 on 6
8.	$u = f(t_1, t_2, \ldots, t_n)$	SL, 2, 4 and 7
9.	B	Subst equals, 8 into 3.

Now it follows easily that, for all expressions B in $\mathbf{B}$, $B \vdash\!\dashv \psi(B)$ in $\mathbf{B}$.

We can now prove the four conditions of Definition **2**.I.1. Using the fact that $\varphi(A)$ is A for every expression A of $\mathbf{A}$, and that both languages contain sentential logic, these can be rewritten: ▶ **2**.I.1

(1) If $\vdash_{\mathbf{A}} A \Rightarrow A'$ then $\vdash_{\mathbf{B}} \varphi(A) \Rightarrow \varphi(A')$.

(2) If $\vdash_{\mathbf{B}} B \Rightarrow B'$ then $\vdash_{\mathbf{A}} \psi(B) \Rightarrow \psi(B')$.

(3) If A is any expression of $\mathbf{A}$, then $\vdash_{\mathbf{A}} A \Leftrightarrow \psi(\varphi(A))$.

(4) If B is any expression of $\mathbf{B}$, then $\vdash_{\mathbf{B}} B \Leftrightarrow \varphi(\psi(B))$.

We know (1) already, (3) is trivial, since $\psi(\varphi(A)) \ominus A$ in this case, and (4) has just been proved above.

It remains to prove (2), which we do by proving that, if $\mathbf{B} \vdash B$, then $\mathbf{A} \vdash \psi(B)$.

Let $L_1, L_2, \ldots, L_n$ be a proof of B in **B**; we show that $\psi(L_1), \psi(L_2), \ldots, \psi(L_n)$, with perhaps some extra lines inserted, forms a proof of $\psi(B)$ in **A**. We observe that L_n is B, so $\psi(L_n) \ominus \psi(B)$.

Suppose the line L arises by Modus Ponens in the old proof. Then there are two previous lines of the form K and $K \Rightarrow L$ in that proof. But then the lines $\psi(K)$ and $\psi(K) \Rightarrow \psi(L)$ occur before $\psi(L)$ in the new proof, and so it arises by Modus Ponens in the new proof also.

Suppose the line L arises by UG in the old proof. Then it is of the form $(\forall x)A(x)$ and there is an earlier line, K say, of the form $A(x)$. But then $\psi(L)$ is $(\forall x)\psi(A(x))$ and follows by UG from $\psi(K)$ which is $\psi(A(x))$ — note that we are talking about the proof of a theorem, not a deduction, here, so there is no proviso regarding the appearance of the variable x in hypotheses to check.

Finally suppose that L is an axiom of **B**. Then there are two possibilities: either it is one of the axioms of **A**, perhaps extended to include the symbol f, or else it is the new axiom. If L is an instance of an Axiom of **A** then $\psi(L)$ is another instance of the same Axiom in **A**.

It remains to consider the possibility that L is the new axiom

$$(\forall u)(u = f(x_1, x_2, \ldots, x_n) \;\Rightarrow\; P(u, x_1, x_2, \ldots, x_n)) \; .$$

This is a single axiom, not an axiom scheme, so the symbol f occurs exactly once and $f(x_1, x_2, \ldots, x_n)$ is a simple f-term. If v is the first variable symbol which does not occur here, we have:

$$\psi(L) \qquad \ominus \qquad (\exists v)P(v, x_1, x_2, \ldots, x_n) \;\wedge\; (\forall u)(\forall v)(u = v \Rightarrow P(u, x_1, x_2, \ldots, x_n)) \; .$$

This is a theorem of **A** (proof below), so all we need do is insert its proof before the line $\psi(L)$ in the new proof.

Here is a proof:

1.	$(\exists!u)P(u, x_1, x_2, \ldots, x_n)$	Assumed to be a theorem
2.	$(\exists u)P(u, x_1, x_2, \ldots, x_n)$	From 1 by SL
3.	$P(v, x_1, x_2, \ldots, x_n)$	Choice
4.	$u = v \;\Rightarrow\; \big(P(v, x_1, x_2, \ldots, x_n) \Rightarrow P(u, x_1, x_2, \ldots, x_n)\big)$	Eq2
5.	$P(v, x_1, x_2, \ldots, x_n) \;\Rightarrow\; \big(u = v \Rightarrow P(u, x_1, x_2, \ldots, x_n)\big)$	SL on 4
6.	$u = v \Rightarrow P(u, x_1, x_2, \ldots, x_n)$	MP, 3 and 5
7.	$(\forall u)(u = v \Rightarrow P(u, x_1, x_2, \ldots, x_n))$	UG on 6
8.	$P(v, x_1, x_2, \ldots, x_n) \wedge (\forall u)(u = v \Rightarrow P(u, x_1, x_2, \ldots, x_n))$	SL, 3 and 7
9.	$(\exists v)\big(P(v, x_1, x_2, \ldots, x_n) \wedge (\forall u)(u = v \Rightarrow P(u, x_1, x_2, \ldots, x_n))\big)$	ExGen on 8 ■

B The Elementary Theory of Arithmetic

Our aim here is to axiomatise elementary arithmetic, so we will need Predicate Logic with Equality and also Peano's axioms. On the other hand we will try to keep it simple and, in particular, avoid introducing set theory.

Peano's axioms, stated informally, are:—

(1) 0 is a natural number

(2) For every natural number n, there is defined a natural number n^+, called the *successor* of n.

(3) 0 is not the successor of any natural number,

(4) The successors of different natural numbers are different.

(5) The only set of natural numbers which contains 0 and is closed under successors is the set of all the natural numbers itself.

(Alternatively: if $P(x)$ is an expression such that $P(0)$ is true and $(\forall x)(P(x) \Rightarrow P(x^+)$, then $(\forall x)P(x)$.)

To set this up as a bona fide theory, we make it a first order theory with equality and add addition and multiplication as functions and axioms as in the following definition.

B.1 Definition

A *first order theory with arithmetic* is a first order theory with equality with the extra function symbols

one constant (nullary function)	$\bar{0}$
one unary function	$x \mapsto x^+$
two binary functions	$+$ and $.$

and the proper axioms

(NT1) $(\forall x)\neg(x^+ = \bar{0})$

(NT2) $(\forall x)(\forall y)\,(x^+ = y^+ \Rightarrow x = y)$

(NT3) Schema: for every expression P and variable symbol x,
$$P(\bar{0}) \;\Rightarrow\; \big((\forall x)(P(x) \Rightarrow P(x^+)) \;\Rightarrow\; (\forall x)P(x)\big)$$

(NT4) $(\forall x)\,(x + \bar{0} = x)$

(NT5) $(\forall x)(\forall y)\,(x + y^+ = (x + y)^+)$

(NT6) $(\forall x)\,(x.\bar{0} = \bar{0})$

(NT7) $(\forall x)(\forall y)\,(x.(y^+) = (x.y) + x)$

Note The theory may contain more functions, relation symbols and axioms than the minimum specified above.

B.2 Remarks

(i) The first three axioms here obviously encapsulate Peano's axioms. I am using $\bar{0}$ for the symbol in the language that stands for zero, allowing us to use 0 for the natural number zero itself.

The third axiom looks a bit more friendly if we rewrite it slightly informally:

$$\big(\ P(\bar{0})\ \wedge\ (\forall x)(P(x) \Rightarrow P(x^+))\ \big) \quad \Rightarrow \quad (\forall x)P(x)$$

Note how it expresses the Principle of Induction without the need for any set theory notation. It is frequently used in the form of a deduction:

$$P(\bar{0})\ ,\ (\forall x)(P(x) \Rightarrow P(x^+)) \ \vdash\ (\forall x)P(x) \qquad \text{(NT3a)}$$

(ii) Since we are avoiding set theory, we have no easy mechanism for introducing new functions. Thus the two basic functions, addition and multiplication, are built in to the definition of the theory itself. Axioms (NT4) and (NT5) constitute an inductive definition of addition, Axioms (NT6) and (NT7) do the same for multiplication.

Herein lies a major shortcoming of this theory — it is difficult to introduce further functions, without extending the language. For example, how can you define exponentiation and prove the index laws in this language?

(iii) We have allowed the possibility of adding extra function and relation symbols and extra axioms. Thus we can approach this theory in two ways. We can consider the bare theory of arithmetic, having only the functions, relations and axioms stated above, and see what we can find out about this particular theory. This theory is usually called the *Elementary Theory of Arithmetic*; we will denote it **ETA**. Alternatively, we can use this definition to discuss any theory which contains arithmetic. This latter approach is useful when we come to Gödel's Theorem, which applies to any theory which contains arithmetic.

(iv) Continuing the ideas of (iii) above. In general, if we take an idea normally defined in mathematics by a set of axiom-like statements, and extract those to form a first-order theory by themselves, we form what is usually called the "Elementary Theory of" whatever it is. We will look briefly at elementary theories of groups and certain kinds of orders later in this chapter.

Such elementary theories do not normally contain the full power of Mathematics — set theoretic notation, functions and so on — so there are usually many things we would like to say about such systems which simply cannot be said, let alone proved, within them.

(v) The language contains terms which describe each of the natural numbers. For instance the term $\bar{0}^+$ represents the number 1, the term $\bar{0}^{++}$ represents the number 2 and so on. For any natural number ξ, let us write $\bar{\xi}$ for the term which represents it — we have already started this off with the term $\bar{0}$. In fact, let us now make this a proper definition.

B.3 Definition

To any natural number ξ, we define a corresponding term $\bar{\xi}$ in **ETA** by induction:

$\bar{0}$ is already given as part of the definition of **ETA**.

$\overline{\xi+1} \ominus \bar{\xi}^{+}$ for all natural numbers ξ.

Thus $\bar{1} \ominus \bar{0}^{+}$, $\bar{2} \ominus \bar{0}^{++}$, $\bar{3} \ominus \bar{0}^{+++}$ and so on.

Note that this notation is not part of the language **S**. It is part of the metalanguage.

B.4 Lemma

$$\vdash \quad (\forall x)(\quad \bar{0} + x = x \quad)$$

Comment This demonstrates how an inductive proof can be constructed in this language. The proof is given in more detail than we would normally bother with.

Why are we proving this when we already have Axiom NT4: $x + \bar{0} = x$? Well, we have not proved commutativity of addition yet, so our theorem does not follow trivially. In fact, this lemma and a similar reversal of NT5 form the basis of a proof of commutativity (see below).

Proof. Here is how we might prove this, using our normal informal language. We will use induction (NT3), so we need to prove

(1) $\bar{0} + \bar{0} = \bar{0}$ and

(2) $(\forall x)(\bar{0} + x = x \ \Rightarrow \ \bar{0} + x^{+} = x^{+})$.

Now (1) is just a case of NT4 and we prove (2) as follows: for any x, suppose that $\bar{0} + x = x$. Then

$$\begin{aligned} \bar{0} + x^{+} &= (\bar{0} + x)^{+} && \text{by NT5} \\ &= x^{+} && \text{by our inductive hypothesis.} \end{aligned}$$

Now let us rewrite this as a semi-formal proof.

1	$\bar{0} + \bar{0} = \bar{0}$	NT4
2	$\bar{0} + x = x$	subhyp
3	$\bar{0} + x^{+} = (\bar{0} + x)^{+}$	NT5
4	$\bar{0} + x^{+} = x^{+}$	Subst of equals (eq3): 1 into 3
5	$\bar{0} + x = x \ \Rightarrow \ \bar{0} + x^{+} = x^{+}$	Ded: 2–4
6	$(\forall x)(\bar{0} + x = x \ \Rightarrow \ \bar{0} + x^{+} = x^{+})$	UG
7	$(\forall x)(\bar{0} + x^{+} = x^{+})$	Induction (NT3a) on 1 and 6

■

This example shows how we can translate an inductive proof, set up in the usual informal way, into a proper semi-formal proof. I will write the proofs more informally from now on.

B.5 Lemma

$$\vdash \quad (\forall x)(\forall y)(\ x^+ + y = (x+y)^+\ )$$

Proof. This is by induction over y. Firstly,

$$\begin{aligned} x^+ + \bar{0} &= x^+ && \text{by NT4} \\ &= (x+\bar{0})^+ && \text{by NT4 again } (x = x+\bar{0} \text{ and substitute equals}). \end{aligned}$$

Now for any y, suppose that $x^+ + y = (x+y)^+$. Then

$$\begin{aligned} x^+ + y^+ &= (x^+ + y)^+ && \text{by NT5} \\ &= (x+y)^{++} && \text{by IH} \qquad (\text{“IH” means “Inductive hypothesis”}) \\ &= (x+y^+)^+ && \text{by NT5 again.} \end{aligned}$$

■

B.6 Theorem

$$\vdash \quad (\forall x)(\forall y)(\ x+y = y+x\ )$$

Proof. This is by induction over y. Firstly

$$x + \bar{0} \ = \ x \ = \ \bar{0} + x \qquad \text{by NT4 and Theorem B4.}$$

Now, for any y, suppose that $x+y = y+x$. Then

$$x + y^+ \ = \ (x+y)^+ \ = \ (y+x)^+ \ = \ y^+ + x \qquad \text{by NT5, IH and Lemma B5.}$$ ■

B.7 Remarks

- It is possible to continue in this vein and prove the usual algebraic properties of addition and multiplication — commutativity, associativity, distributivity, and so on up to prime number factorisation and beyond. I will not follow this up here and now, but restrict myself to a few results which are relevant to the discussion of Gödel's Theorem later.
- While it is difficult to introduce new functions in any useful way into this language (except by formally extending the language by adding new function symbols and axioms), there is no problem with adding new *relations* semiformally. For example, we can define

	$x \neq y$	as an abbreviation for	$\neg(x=y)$
	$x \leq y$	as an abbreviation for	$(\exists u)(x+u=y)$
and	$x < y$	as an abbreviation for	$(\exists u)(x+u^+=y)$.

B.8 Lemma

$$\vdash \quad (\forall x)(\forall y)(\ x<y \Rightarrow x^+ < y^+\ )$$

Proof. From the definition of $<$, there exists u such that $x + u^+ = y$. Then

$$x^+ + u^+ = (x+u^+)^+ = y^+ \qquad \text{and so} \qquad x^+ < y^+ \ .$$ ■

B.9 Lemma

$$\vdash \quad (\forall x)(\ x = \bar{0}\ \vee\ (\exists y)(x = y^+)\ )$$

Proof. is by induction over x. Let $P(x)$ be the statement to be proved. Then $\bar{0} = \bar{0}$ and so $P(\bar{0})$ is true.

Now for any x, $x^+ = x^+$
and so $(\exists y)(y^+ = x^+)$.
Therefore $x^+ = \bar{0}\ \vee\ (\exists y)(y^+ = x^+)$,
in other words, $P(x^+)$.
But then, regardless of the truth of $P(x)$, we have $P(x) \Rightarrow P(x^+)$.
Now, by UG we have $(\forall x)(P(x) \Rightarrow P(x^+))$.
But then, by induction we have $(\forall x)P(x^+)$. ■

B.10 Corollary

$$\vdash \quad (\forall x)(\ x \neq 0 \Rightarrow (\exists y)(x = y^+)\ )$$

(Not so much a corollary as a restatement.)

B.11 Theorem

$$\vdash \quad (\forall x)(\forall y)(\ (x < y)\ \vee\ (x = y)\ \vee\ (y < x)\ )$$

Proof is by induction over x. Suppose first that $x = \bar{0}$. Then either $y = \bar{0}$, in which case $x = y$ and the result is true, or else $y \neq \bar{0}$, in which case by Corollary B10, there is a u such that $y = u^+$. But then $y = x + u^+$, so $x < y$ and again the result is true.

Now we want to prove $(x^+ < y)\ \vee\ (x^+ = y)\ \vee\ (y < x^+)$. Now either $y = \bar{0}$, in which case $x^+ = y + x^+$, so $y < x+$, or else $y \neq \bar{0}$. Then, by Corollary 2.9, there is a u such that $y = u^+$. By IH, $(x < u)\ \vee\ (x = u)\ \vee\ (u < x)$ and now, using Lemma B8, the result follows. ■

B.12 Theorem: The usual full order properties

(i) $\vdash \quad (\forall x)(\ x \leq x\ )$ (Reflexivity)

(ii) $\vdash \quad (\forall x)(\forall y)(\ x \leq y\ \wedge\ y \leq x\ \Rightarrow\ x = y\ )$ (Antisymmetry)

(iii) $\vdash \quad (\forall x)(\forall y)(\forall z)(\ x \leq y\ \wedge\ y \leq z\ \Rightarrow\ x \leq z\ )$ (Transitivity)

(iv) $\vdash \quad (\forall x)(\forall y)(\ x \leq y\ \vee\ y \leq x\ )$ (Dichotomy)

Proof. (i) $x + \bar{0} = x$.

(ii) and (iii) as an interesting exercise.

(iv) follows easily from B11. ■

Question: is it possible in this language to express the fact that the Natural Numbers are well ordered?

B.13 Theorem

$$\vdash \quad (\forall x)(\forall y)(\ x+y=\bar{0}\ \Rightarrow\ x=\bar{0}\ )$$

$$\vdash \quad (\forall x)(\forall y)(\ x+y=\bar{0}\ \Rightarrow\ y=\bar{0}\ )$$

Proof is only given of the second: the proof of the first is similar**.**

We show that $y \neq \bar{0} \vdash x+y \neq \bar{0}$. This is easy. From the hypotheses and Theorem B10, $(\exists u)(y=u^+)$. By the choice rule, $y=a^+$. Then $x+y=x+a^+=(x+a)^+\neq\bar{0}$ by NT5 and NT1. ■

B.14 Lemma

$$\vdash \quad (\forall x)(\quad x=\bar{0}\ \Leftrightarrow\ x\leq\bar{0}\quad)$$

Proof. If $x=\bar{0}$ then $x\leq\bar{0}$; this follows from reflexivity (above).

If $x\leq\bar{0}$ then, by definition of the order relation, there is some u such that $x+u=\bar{0}$. Then by the previous theorem $x=\bar{0}$. ■

B.15 Lemma

(i) $\vdash \quad (\forall x)(\forall y)(\quad x\leq y\ \vee\ x=y^+\ \Leftrightarrow\ x\leq y^+\ )$

(ii) $\vdash \quad (\forall x)(\forall y)(\quad x\leq y\ \Rightarrow\ x\leq y^+\ )$

Proof. (i) Suppose first that $x\leq y$. Then there is a u such that $x+u=y$. Then $x+u^+=(x+u)^+=y^+$ and so $x\leq y^+$. This proves that $x\leq y\ \Rightarrow\ x\leq y^+$. Also $x=y^+\ \Rightarrow\ x\leq y^+$ by reflexivity. Therefore $x\leq y\ \vee\ x=y^+\ \Rightarrow\ x\leq y^+$.

Suppose conversely that $x\leq y^+$. Then there is some u such that $x+u=y^+$. Now either $u=\bar{0}$ or $u\neq\bar{0}$. If $u=\bar{0}$ then $x+\bar{0}=y^+$, that is, $x=y^+$. If $u\neq\bar{0}$ then (by Theorem B10) there is some v such that $u=v^+$ and then we have $(x+v)^+=x+v^+=x+u=y^+$ and so $x+v=y$ by NT2, that is, $x\leq y$. In either case, $x\leq y\ \vee\ x=y^+$ and so we have proved that $x\leq y^+\ \Rightarrow\ x\leq y\ \vee\ x=y^+$.

(ii) This is a corollary of (i). ■

B.16 Theorem

For any natural number η,

$$\vdash \quad (\forall x)(\quad x=\bar{0}\ \vee\ x=\bar{1}\ \vee\ \ldots\ \vee\ x=\bar{\eta}\quad\Leftrightarrow\quad x\leq\bar{\eta}\ .\quad)$$

Note The preamble "for any natural number η" cannot be subsumed into the formula as $(\forall\eta)$, because η is not a variable symbol of the language. Rather, we are stating here a whole set of theorems, one for each natural number η. The first few would be

$$\vdash \ (\forall x)(\ x = \bar{0} \ \Leftrightarrow \ x \leq \bar{0}\ )$$
$$\vdash \ (\forall x)(\ x = \bar{0} \ \vee \ x = \bar{1} \ \Leftrightarrow \ x \leq \bar{1}\ )$$
$$\vdash \ (\forall x)(\ x = \bar{0} \ \vee \ x = \bar{1} \ \vee \ x = \bar{2} \ \Leftrightarrow \ x \leq \bar{2}\ )$$

and so on. One could say that the theorem, as stated, is really a *theorem schema*.

Proof is by (ordinary) induction over the number η**.** For the case $\eta = 0$ we must prove that

$$\vdash \ x = \bar{0} \ \Leftrightarrow \ x \leq \bar{0}$$

and this was our Lemma B14.

Now we suppose that $\vdash \ x = \bar{0} \ \vee \ x = \bar{1} \ \vee \ \ldots \ \vee \ x = \bar{\eta} \ \Leftrightarrow \ x \leq \bar{\eta}$

and prove that $\vdash \ x = \bar{0} \ \vee \ x = \bar{1} \ \vee \ \ldots \ \vee \ x = \overline{\eta+1} \ \Leftrightarrow \ x \leq \overline{\eta+1}$.

By virtue of our inductive assumption and Definition B3, it is enough to prove

$$\vdash \ x \leq \bar{\eta} \ \vee \ x = \bar{\eta}^+ \ \Leftrightarrow \ x \leq \bar{\eta}^+$$

and this follows from Lemma B15(i). ■

B.17 Lemma

For any expression $P(x)$,

$$\vdash \ P(\bar{0}) \ \Leftrightarrow \ (\forall x \leq \bar{0})P(x)\,.$$

[Here we are introducing the notation $(\forall x \leq t)P(x)$ as an (obvious?) abbreviation for $(\forall x)(x \leq t \Rightarrow P(x))$.]

Proof. First we prove

$$P(\bar{0})\ ,\ x \leq \bar{0} \ \vdash \ P(x)\,. \tag{1}$$

From the hypothesis $x \leq \bar{0}$ and Lemma B14 we get $x = \bar{0}$. Then we get $P(x)$ from $P(\bar{0})$ by substitution of equals.

From (1) by the Deduction Theorem we have $P(\bar{0}) \vdash x \leq \bar{0} \Rightarrow P(x)$

and UG gives $P(\bar{0}) \vdash (\forall x)(x \leq \bar{0} \Rightarrow P(x))$

which is the same as $P(\bar{0}) \vdash (\forall x \leq \bar{0})P(x)$.

By the Deduction Theorem once more,

$$\vdash \ P(\bar{0}) \Rightarrow (\forall x \leq \bar{0})P(x)\,. \tag{2}$$

Next we prove

$$(\forall x \leq \bar{0})P(x) \ \vdash \ P(\bar{0})\,, \tag{3}$$

that is $$(\forall x)(x \leq \bar{0} \Rightarrow P(x)) \quad \vdash \quad P(\bar{0})\,.$$

From the hypothesis by PL6, $\bar{0} \leq \bar{0} \Rightarrow P(\bar{0})$. But $\bar{0} \leq \bar{0}$ by reflexivity. Now MP gives $P(\bar{0})$.

Applying the Deduction Theorem to (3) gives $\vdash (\forall x \leq \bar{0})P(x) \Rightarrow P(\bar{0})$ which with (2) gives the lemma. ■

B.18 Lemma

For any expression $P(x)$,

$$\vdash \quad (\forall x \leq y)P(x) \ \wedge \ P(y^+) \quad \Leftrightarrow \quad (\forall x \leq y^+)P(x)$$

Proof. First we prove

$$(\forall x \leq y)P(x) \ \wedge \ P(y^+)\ ,\ x \leq y^+ \quad \vdash \quad P(x)\,. \tag{1}$$

From the first hypothesis we have $(\forall x)(x \leq y \Rightarrow P(x))$ and so, by PL6, $x \leq y \Rightarrow P(x)$. Also from the hypothesis $x \leq y^+$ by Lemma B15(i) we have either $x \leq y$ or $x = y^+$. If $x \leq y$ then MP gives $P(x)$. If $x = y^+$ then substitution of equals in $P(y^+)$ from the first hypothesis gives $P(x)$.

From (1) by the Deduction Theorem we have

$$(\forall x \leq y)P(x) \ \wedge \ P(y^+) \quad \vdash \quad x \leq y^+ \Rightarrow P(x)\,.$$

Then UG gives

$$(\forall x \leq y)P(x) \ \wedge \ P(y^+) \quad \vdash \quad (\forall x)(x \leq y^+ \Rightarrow P(x))\,,$$

that is

$$(\forall x \leq y)P(x) \ \wedge \ P(y^+) \quad \vdash \quad (\forall x \leq y^+)P(x)\,. \tag{2}$$

Next we prove

$$(\forall x \leq y^+)P(x)\ ,\ x \leq y \quad \vdash \quad P(x)\,. \tag{3}$$

The first hypothesis is $(\forall x)(x \leq y^+ \Rightarrow P(x))$ which, with PL6, gives $x \leq y^+ \Rightarrow P(x)$. The second hypothesis with Lemma B15(ii) gives $x \leq y^+$. Now MP gives $P(x)$.

As before, applying the Deduction Theorem and UG to (3) gives

$$(\forall x \leq y^+)P(x) \quad \vdash \quad (\forall x \leq y)P(x)\,. \tag{4}$$

Next we note that

$$(\forall x \leq y^+)P(x) \quad \vdash \quad P(y^+)\,. \tag{5}$$

This is because the hypothesis, which is $(\forall x)(x \leq y^+ \Rightarrow P(x))$, gives $y^+ \leq y^+ \Rightarrow P(y^+)$ by PL6. But $y^+ \leq y^+$ by reflexivity and so MP gives (5).

From (4) and (5) now we have $(\forall x \leq y^+)P(x) \vdash (\forall x \leq y)P(x) \ \wedge \ P(y^+)$ and so

$$\vdash (\forall x \leq y^+)P(x) \ \Rightarrow \ (\forall x \leq y)P(x) \ \wedge \ P(y^+)\,.$$

This, with (2), gives the lemma. ■

B.19 Theorem

For any expression $P(x)$ and any natural number η,

$$\vdash \quad P(\bar{0}) \ \wedge \ P(\bar{1}) \ \wedge \ \ldots \ \wedge \ P(\bar{\eta}) \quad \Leftrightarrow \quad (\forall x \leq \bar{\eta})P(x)\,.$$

Proof is by (ordinary) induction over η. In the case $\eta = 0$, we must prove that

$$\vdash \quad P(\bar{0}) \quad \Leftrightarrow \quad (\forall x \leq \bar{0})P(x)\,.$$

But this is Lemma B17.

Now suppose that

$$\vdash \quad P(\bar{0}) \ \wedge \ P(\bar{1}) \ \wedge \ \ldots \ \wedge \ P(\bar{\eta}) \quad \Leftrightarrow \quad (\forall x \leq \bar{\eta})P(x)$$

and we want to prove that

$$\vdash \quad P(\bar{0}) \ \wedge \ P(\bar{1}) \ \wedge \ \ldots \ \wedge \ P(\overline{\eta+1}) \quad \Leftrightarrow \quad (\forall x \leq \overline{\eta+1})P(x)\,.$$

By virtue of the inductive hypothesis and Definition B3, it is enough to prove that

$$\vdash \quad (\forall x \leq \bar{\eta})P(x) \ \wedge \ P(\bar{\eta}^{+}) \quad \Leftrightarrow \quad (\forall x \leq \bar{\eta}^{+})P(x)$$

and this is given by Lemma B18. ■

C The Elementary Theory of Groups

C.1 The theory

The *Elementary theory of groups* is a first order theory with equality, plus three function symbols:

one nullary,	e
one unary,	n
one binary,	m

and the following proper axioms:—

(G1) $m(x, m(y, z)) = m(m(x, y), z)$

(G2) $m(x, e) = x$

(G3) $m(x, n(x)) = e$.

If we employ a more usual notation,

e	is the identity 1
$n(x)$	is the inverse x^{-1}
$m(x, y)$	is the product xy

then the axioms look more familiar:

(G1) $x(yz) = (xy)z$

(G2) $x1 = x$

(G3) $xx^{-1} = 1$.

C.2 Remarks

(i) This is a stripped-down version of the axioms. In particular we do not assert

	(G2′)	$1x = x$
or	(G3′)	$x^{-1}x = 1$

as axioms. These can be proved as theorems. To find the proofs is an interesting exercise (hint: prove G3′ first).

(ii) This theory cannot encompass the whole of Group Theory as we know it. As it is set up, all variables must be elements of the same group and so the theory can only talk about one group at a time (a single group constitutes the whole "universe of discourse"). Thus the theory is not categorical, in the sense of the question posed at the end of 1.A1. This will be made precise in the Chapter 6.

This also means that the theory cannot talk about homomorphisms, subgroups, direct products etc. etc.

C.3 Alternative axiomatisations

There are many other (equivalent) ways of axiomatising the Elementary Theory of Groups. I will only mention one here, because it corresponds to what is perhaps the most common way of defining groups in introductory texts. In this form, there is only one function, multiplication, and the axioms change to

(GA1) $x(yz) = (xy)z$

(GA2) $(\exists e)(\forall x)\big(xe = x \ \wedge \ (\exists y)(xy = e)\big)$.

In developing the theory from this base, the first task is to show that the elements e and y mentioned in GA2 are in fact unique. Then they give rise to the identity and inverse as functions defined by description.

D Unbounded dense linear orders (UDLO)

D.1 The theory

The *Elementary Theory of Unbounded Dense Linear Orders* (**UDLO**) is a first order theory with equality, no function symbols and one extra relation $<$ which will (semi-formally) be written in infix notation, as is usual. It has the following proper axioms:—

(UDLO1) $(\forall x)(\forall y)(\ x < y \ \vee \ x = y \ \vee \ y < x\ )$

(UDLO2) $(\forall x)(\forall y)\neg(\ x < y \ \wedge \ y < x\ )$

(UDLO3) $(\forall x)(\forall y)(\forall z)(\ x < y \ \wedge \ y < z \ \Rightarrow \ x < z\ )$

(UDLO4) $(\forall x)(\forall y)(\ x < y \ \Rightarrow (\exists z)(x < z \ \wedge \ z < y)\ )$

(UDLO5) $(\forall x)(\ (\exists z)(z < x) \ \wedge \ (\exists z)(x < z)\ )$

D.2 Remarks

(i) Axioms UDLO1–UDLO3 express the fact that the system is fully ordered, in terms of a strict order relation $<$. Axiom UDLO4 expresses *density*: between any two distinct elements there is another. Axiom UDLO5 says that the system is unbounded: it has neither a least nor a greatest element.

(ii) In our semi-formal exposition we allow the usual notation, $\neq$, $>$, $\leq$ and $\geq$, defined in the usual way.

(iii) In what follows it will be very convenient to allow the symbols **T** and **F** for "true" and "false". We can either add these as part of the logical background or add **T** as a new nullary relation together with defining axiom

$$\mathbf{T}$$

and then define **F** as a semi-formal abbreviation for $\neg\mathbf{T}$.

D.3 Theorem

UDLO is not categorical.

Proof. The Rationals $\mathbb{Q}$ and the Reals $\mathbb{R}$ are both systems described by **UDLO**; they are clearly not isomorphic. ■

D.4 Comment

The proof above is a wee bit vague, due mainly to the slight vagueness in our definition of "categorical". What we are really saying here is that $\mathbb{Q}$ and $\mathbb{R}$ are non-isomorphic *models* of **UDLO**. In Chapter **7** we will define all these terms properly and this theorem and its proof will become watertight.

► Ch.7

The remainder of this section is devoted to a proof that **UDLO** is complete. Throughout, the symbol $\vdash$ means, of course, proveable in **UDLO**.

D.5 Lemma

Let $x_1, x_2, \ldots, x_n, z$ be variable symbols ($n \geq 1$). Then

$$\vdash \ (\exists z)\big(\ x_1 \leq z \ \wedge\ x_2 \leq z \ \wedge\ \ldots\ \wedge\ x_n \leq z \ \wedge\ (x_1 = z \ \vee\ x_2 = z \ \vee\ \ldots\ \vee x_n = z)\ \big)\,. \tag{1}$$

This expresses the idea that the set $\{x_1, x_2, \ldots, x_n\}$ has a maximum member, using the language available to us. In the same way we can express the fact that the set has a minimum member.

$$\vdash \ (\exists z)\big(\ z \leq x_1 \ \wedge\ z \leq x_2 \ \wedge\ \ldots\ \wedge\ z \leq x_n \ \wedge\ (z = x_1 \ \vee\ z = x_2 \ \vee\ \ldots\ \vee z = x_n)\ \big)\,. \tag{2}$$

Proof. We prove the first statement only, by induction over n. In the case $n = 1$,

$$\vdash \ (\exists z)(\ x_1 \leq z \ \wedge\ x_1 = z\ )$$

is obvious (set $z \ominus x_1$).

Now in the case $n \geq 2$, and slipping into more informal style, we assume as inductive hypothesis that there exists z' such that

$$x_1 \leq z' \ \wedge\ x_2 \leq z' \ \wedge\ \ldots\ \wedge\ x_{n-1} \leq z' \ \wedge\ (x_1 = z' \ \vee\ x_2 = z' \ \vee\ \ldots\ \vee x_{n-1} = z')\,. \tag{3}$$

Now we have two possibilities: either

$$z' \leq x_n \tag{4}$$

or

$$x_n \leq z'\,. \tag{5}$$

In the first case, set

$$z \ominus x_n \quad \text{so that} \quad z' \leq z \tag{6}$$

and we have $x_1 \leq z \ \wedge\ x_2 \leq z \ \wedge\ \ldots\ \wedge\ x_{n-1} \leq z$ from (3) and (6). Also $z = x_n$, so we have

$$\vdash \ x_1 \leq z \ \wedge\ x_2 \leq z \ \wedge\ \ldots\ \wedge\ x_n \leq z \ \wedge\ (x_1 = z \ \vee\ x_2 = z \ \vee\ \ldots\ \vee x_n = z)\,.$$

and (1) follows. In the second case, set

$$z \ominus z' \tag{7}$$

and we have

$$\vdash \ x_1 \leq z \ \wedge\ x_2 \leq z \ \wedge\ \ldots\ \wedge\ x_{n-1} \leq z \ \wedge\ (x_1 = z \ \vee\ x_2 = z \ \vee\ \ldots\ \vee x_{n-1} = z)$$

from (3) and (7). ■

You will notice that we have used both the choice Rule and the Constructive Dilemma in this proof.

D.6 Note

The atomic expressions of **UDLO** are those of one of the forms

$$\mathsf{T} \quad , \quad x = y \quad or \quad x < y$$

where x and y are variable symbols. By *simple expressions* we will mean expressions which are either atomic expressions or negations of atomic expressions, so these are expressions of one of the forms

$$\mathsf{T} \quad , \quad \neg T \quad , \quad x = y \quad , \quad \neg(x = y) \quad , \quad x < y \quad \text{or} \quad \neg(y < x) \; .$$

These are the same, or equivalent to, expressions of one of the forms

$$\mathsf{T} \quad , \quad \mathsf{F} \quad , \quad x = y \quad , \quad x \neq y \quad , \quad x < y \quad \text{or} \quad x \leq y \; .$$

By a *C-term* we will mean an expression of the form

$$A_1 \; \wedge \; A_2 \; \wedge \; \ldots \; \wedge \; A_m$$

where each A_i is an atomic expression.

By a *DC-form* we will mean an expression which is either F or of the form

$$B_1 \; \vee \; B_2 \; \vee \; \ldots \; \vee \; B_n$$

were each B_i is a C-term.

D.7 Lemma

Let P be any expression in **UDLO** which contains no quantifiers. Then $\vdash P \Leftrightarrow Q$, where Q is a DC-form which mentions no variable symbols which do not occur in P.

Proof. If P is atomic then this is so: look at the cases.

$$\begin{aligned} &\vdash \quad \neg\mathsf{T} \Leftrightarrow \mathsf{F} \\ &\vdash \quad \neg\mathsf{F} \Leftrightarrow \mathsf{T} \\ &\vdash \quad \neg(x < y) \Leftrightarrow (y < x) \vee (y = x) \\ &\vdash \quad \neg(x = y) \Leftrightarrow (x < y) \vee (y < x) \end{aligned}$$

Now suppose that P is a C-term, $A_1 \wedge A_2 \wedge \cdots \wedge A_m$ say. Then

$$\vdash \quad P \Leftrightarrow \neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_m$$

and, as proved above, for each i, $\vdash \neg A_i \Leftrightarrow B_i$, where B_i is a DC-form, so

$$\vdash \quad P \Leftrightarrow B_1 \vee B_2 \vee \cdots \vee B_m \, ,$$

which itself a DC-form.

Finally, suppose that P is an arbitrary DC-form, $B_1 \vee B_2 \vee \cdots \vee B_n$ say. Then

$$\vdash \quad P \Leftrightarrow \neg B_1 \wedge \neg B_2 \wedge \cdots \wedge \neg B_n \,.$$

Then, as just proved, for each i there is a DC-form D_i such that $\vdash \neg B_i \Leftrightarrow D_i$ and so

$$\vdash \quad P \Leftrightarrow D_1 \wedge D_2 \wedge \cdots \wedge D_n \,. \tag{1}$$

Write out each D_i as $C_{i,1} \wedge C_{i,2} \wedge \cdots \wedge C_{i,m_i}$, where the $C_{i,j}$ are C-terms, substitute in (1) and expand by the distributive law; the result is a DC-form. ■

D.8 Lemma

Let P be any expression in **UDLO** which contains no quantifiers. Then

$$\vdash \quad (\exists z)P \;\Leftrightarrow\; Q$$

where Q is a DC-form which does not mention z and mentions no variable symbols which do not occur in P.

Proof. By the previous lemma we may assume that P is a DC-form, that is,

$$P \quad \ominus \quad B_1 \vee B_2 \vee \cdots \vee B_n$$

where each B_i is a C-term. But then (by Predicate Logic)

$$(\exists z)P \quad \ominus \quad ((\exists z)B_1) \vee ((\exists z)B_2) \vee \cdots \vee ((\exists z)B_n)$$

so it is sufficient to prove the result in the case that P is in fact a C-term. So let us assume that

$$P \quad \ominus \quad A_1 \wedge A_2 \wedge \cdots \wedge A_m \,,$$

where the A_i are all atomic expressions (let's call them its *factors*). The proof is by induction over m, the number of factors in P.

Firstly note that, if none of the factors A_i mention z then the result is trivially true with Q the same as P. So from now on we may assume that at least one of the factors mentions z.

Next observe that if any one of the factors A_i does not mention z, then it may be "taken outside"; for instance, if A_1 does not mention z, then

$$\vdash \quad (\exists z)(A_1 \wedge A_2 \wedge \cdots \wedge A_n) \quad \Leftrightarrow \quad A_1 \;\wedge\; (\exists z)(A_2 \wedge \cdots \wedge A_n)$$

and then the inductive hypothesis gives the result. So from now on we may assume that *all* the factors $A_1, A_2, \ldots, A_n$ mention z.

Now the factors of P are each of one of the following forms (where here x stands for any variable symbol which is different from z).

$$z = x \quad , \quad z < x \quad , \quad x < z \quad , \quad z = z \quad , \quad z < z \,.$$

If P contains a factor of the form $z = z$, then that factor may be simply eliminated, since it is equivalent to $\mathbf{T}$, and then the inductive hypothesis applies. If P contains a factor of the form $z < z$, then $P \Leftrightarrow \mathbf{F}$ since that factor is equivalent to $\mathbf{F}$.

Suppose now that P contains a factor of the form $z = x$. Rearranging the factors if necessary, P is equivalent to an expression of the form $(z = x) \ \wedge \ R(z)$, where $R(z)$ is the conjunction of all the remaining factors. (I have made the variable z explicit here because we are going to substitute for it.) Now

$$\vdash \ (\exists z)P \ \Leftrightarrow \ (\exists z)\big((z = x) \wedge R(z)\big) \quad \text{and} \quad \vdash \ (\exists z)\big((z = x) \wedge R(z)\big) \Leftrightarrow \ R(x)$$

and $R(x)$ is a conjunction of $m - 1$ atomic expressions, so again the inductive hypothesis applies.

From now on we may assume that all of the factors of P are of one of the forms $z < x$ and $x < z$.

Suppose now that P contains two such factors with the same variable x. If the two factors are in fact identical, then they can be replaced by one copy and the inductive hypothesis applies. On the other hand, if the two factors are different, they must be of the form $z < x$ and $x < z$, with the same x. But

$$\vdash \ z < x \ \wedge \ x < z \quad \Leftrightarrow \quad \mathbf{F}$$

and then $\vdash (\exists z)P \Leftrightarrow F$, as required.

From now on we may assume that the variable symbol x which turns up in each factor of P is different. We now need to write P out more explicitly. There are variable symbols

$$x_1, x_2, \ldots, x_p \quad , \quad y_1, y_2, \ldots, y_q \ ,$$

all distinct, such that

$$P \quad \ominus \quad (x_1 < z) \wedge (x_2 < z) \wedge \cdots \wedge (x_p < z) \ \wedge \ (z < y_1) \wedge (z < y_2) \wedge \cdots \wedge (y_q < z) \qquad (1)$$

I will prove that then $(\exists z)P$ is equivalent to the expression Q which is the conjunction of all the simple expressions

$$x_i < y_j \quad \text{for} \quad i = 1 \ldots p \quad \text{and} \quad j = 1 \ldots q \qquad (2)$$

(if there are no such expressions at all, we take this to mean that Q is $\mathbf{T}$).

It is not difficult to see that, in ordinary mathematics, (1) and (2) are equivalent (the proof is "Stare hard"). The point here is to check that they are proveably equivalent with only the limited resources of **UDLO**.

To see that $\vdash (\exists z)P \Rightarrow Q$ we argue as follows. For any i and j, $x_i < z$ and $z < y_j$ are factors of P, so $\vdash P \ \Rightarrow \ x_i < z \ \wedge \ z < y_j$ and so $\vdash P \ \Rightarrow \ x_i < y_j$ and thus $\vdash (\exists z)P \ \Rightarrow \ x_i < y_j$. Repeating this for each i and j, we see that $(\exists z)P$ implies every one of the factors of Q listed above and so it implies their conjunction, which is Q.

It remains to show that $\vdash Q \Rightarrow (\exists z)P$.

Suppose first that P has no terms of the form $z < y_j$; Then Q is T so we want to prove that $\vdash T \Rightarrow (\exists z)P$, that is, that $\vdash P$. By the previous lemma (D.5),

$$\vdash \quad (\exists z)\big(\ x_1 \le z \ \wedge\ x_2 \le z \ \wedge\ \ldots\ \wedge\ x_p \le z \ \wedge \\ (\ z = x_1 \ \vee\ z = x_2 \ \vee\ \ldots \vee z = x_p)\ \big)\,. \tag{3}$$

Using a choice argument, we choose such a z, and so deduce that $\vdash x_i \le z$ for all i. Using the axiom UDLO4, there is z' such that $z < z'$, from which we deduce $\vdash x_i < z'$ for all the x_i, and then $\vdash (\exists z)P$ follows immediately.

A similar argument proves the lemma in the case in which P has no terms of the forms $x_i < z$.

Finally, we assume that P has at least one term of the form $x_i < z$ and at least one term of the form $z < y_j$. For this we need both forms of the previous lemma: there are theorems of the form (3) above and

$$\vdash \quad (\exists z)\big(\ z \le y_1 \ \wedge\ z \le y_2 \ \wedge\ \ldots\ \wedge\ z \le y_q \ \wedge \\ (\ z = y_1 \ \vee\ z = y_2 \ \vee\ \ldots \vee z = y_q)\ \big)\,. \tag{4}$$

Using the Choice Rule (and changing letters for convenience), there are theorems of the forms

$$\vdash \quad x_i \le u \tag{5}$$

$$\vdash \quad u = x_1 \ \vee\ u = x_2 \ \vee\ \ldots \ \vee\ u = x_p \tag{6}$$

$$\vdash \quad v \le y_j \tag{7}$$

$$\vdash \quad v = y_1 \ \vee\ v = y_2 \ \vee\ \ldots \ \vee\ v = y_q \tag{8}$$

((5) is a list of theorems, one for each $i = 1, 2, \ldots, p$, and (7) is a list of theorems, one for each q.)

Applying Proof by Cases (the Constructive dilemma) to (6) and (8), we prove a heap of special cases, one for each i and j, as follows.

For any particular i and j, we have a case $u = x_i \ \wedge\ v = y_j$, and we use (2) to get $\vdash u < v$. Then the axiom UDLO4 tells us that $\vdash (\exists z)(u < z < v)$ and then, using (5) and (7), for each i' and j',

$$\vdash \quad x_{i'} < z$$
$$\vdash \quad z < y_{j'}$$

which immediately gives $(\exists z)P$, as required. ■

The next lemma removes the proviso that P contains no quantifiers from the last one.

D.9 Lemma

Let P be any expression in **UDLO**. Then

$$\vdash \quad P \ \Leftrightarrow\ Q$$

where Q is a DC-form which does not mention z and mentions no variable symbols which do not occur free in P.

Proof. We proceed by induction over the construction of P.

If P is an atomic expression, then it is already of the required form.

If P is of the form $\neg R$, then we may assume inductively that $\vdash R \Leftrightarrow R'$, where R' is a DC-form. But then $\vdash P \Leftrightarrow \neg R'$ and $\neg R'$ is an expression with no quantifiers, so by Lemma
▶ D.7 D.7 there is a DC-form Q such that $\vdash \neg R' \Leftrightarrow Q$.

Following this argument through, observe that R mentions the same variables as does P, so R' mentions no variables that do not occur in P and then the same is true of Q.

The same argument can be used when P is of the form $R \Rightarrow S$

Finally, if P is $(\forall z)R$, we may again assume inductively that $\vdash R \Leftrightarrow R'$, where R' is a
▶ D.7 DC-form and then (by Lemma D.7) that $\vdash \neg R \Leftrightarrow R''$ where R'' is another DC-form. But
▶ D.8 now $\vdash (\exists z)\neg R \Leftrightarrow (\exists z)R''$ and so, by Lemma D.8, $\vdash (\exists z)\neg R \Leftrightarrow R'''$, where R''' is yet another DC-form. Finally, $\vdash P \Leftrightarrow \neg(\exists z)\neg R$ so $\vdash P \Leftrightarrow \neg R'''$ and then, by Lemma D.7 again, $\vdash P \Leftrightarrow Q$, another DC-form.

Once again, following this argument through, the variables which occur free in R are a subset of those which occur free in P, together possibly with z. Then R' mentions a subset of those same variables and R'' mentions a subset of that.. Then R''' mentions a subset of that, with the exception of z, and that is a subset of the variables which occur free in P. Finally Q mentions a subset of that, as required.

■

D.10 Corollary

UDLO is complete.

Proof. Let P be any sentence (closed expression) in **UDLO**. By the lemma, $\vdash P \Leftrightarrow Q$, where Q is an expression which has no quantifiers and no free variables; it follows that Q is an expression which is built up entirely from T and sentential logic symbols (so it could be T or something more complicated like $(\mathsf{T} \wedge \neg\mathsf{T}) \Rightarrow (\neg\mathsf{T} \Rightarrow \mathsf{T})$). It is easy to check that any such expression must be equivalent to either T or F. ■

5. VBG SET THEORY

A Von Neumann–Bernays–Gödel Axioms for Set Theory

In this chapter I expound the von Neumann-Bernays-Gödel axioms for Set Theory. This first order theory is sufficiently powerful to form a basis for all mathematics (if we exclude branches which investigate what can be done with other logical systems, such as intuitionistic mathematics and metamathematics, as we must).

While it is obviously impossible to actually develop the whole of mathematics explicitly here, I will provide the beginning of this program, enough so that I hope it will become obvious that it can indeed be carried through.

I will start by defining the theory, presenting the axioms without comment — some of them may be difficult to understand at this stage. They will be explained as they become relevant in the development.

In this chapter we define and discuss the first-order system ***VBG****. Every expression, term and symbol discussed will be part of this language, and no complicated meta-arguments will be engaged in. Consequently, the use of the coloured font to distinguish carefully between the formal language and the metalanguage is not necessary here. I will not bother with font-colouring in this chapter and the next, otherwise almost everything would be in colour.*

A.1 Definition: Von Neumann-Bernays-Gödel Set Theory, VBG

This is a first order theory with equality. It has one binary relation $\in$ and no functions. For simplicity I will use abbreviations for a couple of predicates which occur in these axioms:

$$\begin{array}{rll} \mathrm{SET}(x) & \text{will mean} & (\exists s)(x \in s) \\ x \subseteq y & \text{will mean} & (\forall u)(\ u \in x\ \Rightarrow\ u \in y\ )\ . \end{array}$$

Read the first one as saying "x is a set".

The proper axioms are:—

(VBG1, Extension) $\quad (\forall x)(x \in a \Leftrightarrow x \in b) \quad \Rightarrow \quad a = b$

(VBG2, Specification) Schema: an axiom for every expression $P(x, y_1, y_2, \ldots, y_n)$
$$(\forall y_1)(\forall y_2)\ldots(\forall y_n)(\exists w)(\forall x)(x \in w \quad \Leftrightarrow \quad \mathrm{SET}(x) \wedge P(x, y_1, y_2, \ldots, y_n))$$

(VBG3, Unordered Pairs)
$$\mathrm{SET}(a) \wedge \mathrm{SET}(b) \quad \Rightarrow \quad (\exists w)\big(\mathrm{SET}(w)\ \wedge\ a \in w\ \wedge\ b \in w\big)$$

(VBG4, Unions) $\quad \mathrm{SET}(a) \quad \Rightarrow \quad (\exists w)\big(\mathrm{SET}(w)\ \wedge\ (\forall x)((\exists y)(x \in y\ \wedge\ y \in a)\ \Rightarrow\ x \in w)\big)$

(VBG5, Power Set) $\quad \mathrm{SET}(a) \;\Rightarrow\; (\exists w)\big(\mathrm{SET}(w) \wedge (\forall x)(x \subseteq a \;\Rightarrow\; x \in w)\big)$

(VBG6, Infinity)
$(\exists w)\big(\mathrm{SET}(w) \wedge (\exists u)(u \in w \wedge (\forall x)\neg(x \in u))$
$\wedge\, (\forall x)\big(x \in w \;\Rightarrow\; (\exists y)(y \in w \wedge (\forall u)(u \in y \Leftrightarrow u \in x \vee u = x))\big)\big)$

(VBG7, Formation) Schema: an axiom for every expression $P(x, y)$
$\mathrm{SET}(a) \wedge (\forall x)(x \in a \Rightarrow (\exists! y)P(x, y)) \wedge (\forall x)(\forall y)(x \in a \wedge P(x, y) \Rightarrow \mathrm{SET}(y))$
$\Rightarrow \quad (\exists w)\big(\mathrm{SET}(w) \wedge (\forall y)(y \in w \Leftrightarrow (\exists x)(x \in a \wedge P(x, y)))\big)$

(VBG8, Foundation)
$(\forall a)\big(\,(\mathrm{SET}(a) \wedge a \subseteq w) \;\Rightarrow\; a \in w\,\big) \quad \Rightarrow \quad (\forall a)(\mathrm{SET}(a) \;\Rightarrow\; a \in w)$

Note the Axiom of Choice is *not* an axiom of **VBG**. It will be discussed below in Section F.

A.2 Remark

In **VBG** Set Theory all mathematical objects are classes. A set is defined as a class which is a member of some other class (hence the notation $\mathrm{SET}(x)$ defined above). Thus numbers, functions, relations and so on are all actually constructed as classes — we will see how this comes about — and of course many of these things turn out to be sets. Nevertheless, there are many objects (for examples functions, real numbers) which we don't *think* about as classes. Some of the notation becomes a little clearer if we use uppercase letters for variables which we think of as classes or sets (that is, whose members are of interest to us) and lowercase letters for the others. I will do this where appropriate.

In this chapter, many statements will be made without proof; in all these cases the proofs, I believe, are straightforward.

B The first five axioms

B.1 VBG1: The Axiom of Extension

First observe that the opposite implication, $A = B \quad \Rightarrow \quad (\forall x)(x \in A \Leftrightarrow x \in B)$ is given to us by Substitution of Equals, and so this axiom tells us that

$$A = B \quad \text{if and only if} \quad (\forall x)(x \in A \Leftrightarrow x \in B)\ ,$$

in other words, two classes are equal if and only if they have the same members.

B.2 VBG2: The Axiom of Specification and the Empty Set

The statement of this axiom looks rather complicated. It could have been written this way: for every expression P,

$$(\exists w)(\forall x)(x \in w \quad \Leftrightarrow \quad \mathrm{SET}(x) \wedge P)\,,$$

with the reader being left to realise that the expression P probably mentions the variable x free, and might also have other free variables $y_1, y_2, \ldots, y_n$, say, which would then be free variables in the axiom. On general grounds, it is better not to have free variables in axioms, so this axiom is stated above with all variables bound.

Is there any guarantee that there are any classes at all? Well, yes. Substitute any predicate you like into the Axiom of Specification, for instance $x = x$, and we have

$$(\exists w)(\forall x)(x \in w \quad \Leftrightarrow \quad \mathrm{SET}(x) \wedge x = x)$$

which tells us that there is a class (w) with certain properties, which we won't bother about just now. Suffice it to know that a class actually exists.

Well then, are there any sets at all? Again yes. A glance at the Axiom of Infinity (VBG6) shows us that it states that a set (w again) exists with certain complicated-looking properties. For the moment, that is enough: there exists a set.

Now let us use the Axiom of Specification once more with the predicate $\neg\,\mathrm{SET}(x)$:

$$(\exists w)(\forall x)(x \in w \quad \Leftrightarrow \quad \mathrm{SET}(x) \wedge \neg\,\mathrm{SET}(x))$$

Since $\mathrm{SET}(x) \wedge \neg\,\mathrm{SET}(x)$ is clearly false for all x, this is the same as

$$(\exists w)(\forall x)\neg(x \in w)$$

It is not difficult to prove that this is in fact unique existence, that is,

$$(\exists! w)(\forall x)\neg(x \in w)$$

(To see this, we must prove that, for any w and w',

$$(\forall x)\neg(x \in w) \ \wedge \ (\forall x)\neg(x \in w') \ \Rightarrow \ w = w'$$

and this follows easily from the Axiom of Extension.) This unique existence means that we can define a constant (nullary function) by description; this is of course the Empty Set $\varnothing$. It is thus defined by its property

$$(\forall x)\ \neg(x \in \varnothing)\ .$$

This property simply states that the empty set has no members. Note that we have also established that it is the only set with these properties.

However not so fast: we have not proved yet that $\varnothing$ is in fact a set: its definition above simply guarantees its existence as a class (and we have established that it is the only class with these properties).

Look ahead once more at the Axiom of Infinity. It tells us that there exists a set w with certain properties, of which the first two are $\mathrm{SET}(w)$ and $(\exists u)(u \in w \ \wedge \ (\forall x)\neg(x \in u))$. We now see that the second of these properties says simply that $\varnothing \in w$ which, with the definition of "set" above, tells us that $\varnothing$ is indeed a set.

Given any predicate $P(x)$, then there is a class B such that

$$x \in B \quad \Leftrightarrow \quad \mathrm{SET}(x) \wedge P(x)$$

that is to say, there is a class B which consists of just those sets x for which $P(x)$ is true.

It is easy to prove that B is uniquely defined by P, so we may use functional notation and write

$$\{\, x : P(x) \,\}$$

for this class B. Thus we have the basic way of dealing with such a class:

$$y \in \{\, x : P(x) \,\} \qquad \text{if and only if} \qquad \mathrm{SET}(y) \wedge P(y)$$

In many cases the predicate $P(x)$ will be such that $(\forall x)(P(x) \Rightarrow \mathrm{SET}(x))$ and in such cases

$$y \in \{\, x : P(x) \,\} \qquad \text{if and only if} \qquad P(y)$$

Another result which is easily proved is

$$\text{for any class } A, \qquad A = \{\, x : x \in A \,\}\ .$$

B.3 Some useful notation (definitions)

$x \notin A$	means	$\neg(x \in A)$
$A \subseteq B$	means	$(\forall x)(x \in A \Rightarrow x \in B)$
$A \subset B$	means	$A \subseteq B \quad \wedge \quad A \neq B$
$A \supseteq B$	means	$B \subseteq A$
$A \supset B$	means	$B \subset A$
$(\forall x \in A)P(x)$	means	$(\forall x)(x \in A \Rightarrow P(x))$
$(\exists x \in A)P(x)$	means	$(\exists x)(x \in A \wedge P(x))$

Note that all these definitions hold whenever A and B are classes — there is no requirement that they be sets. With regard to the last two notations, observe that if the set A happens to be empty, $(\forall x \in A)P(x)$ is automatically true regardless of the expression $P(x)$. We say that the statement is *vacuously true*. Similarly, if A is empty, the statement $(\exists x \in A)P(x)$ is automatically false.

B.4 Properties of subclasses

The following are now easily proved

(i) The empty set is a subset of every class:

$$\varnothing \subseteq A$$

and the only subclass of the empty set is itself:

$$A \subseteq \varnothing \quad \Leftrightarrow \quad A = \varnothing$$

(ii) The subclass relation has the properties of a partial order:

Reflexivity	$A \subseteq A$					
Antisymmetry	$A \subseteq B$	&	$B \subseteq A$	$\Rightarrow$	$A = B$	
Transitivity	$A \subseteq B$	&	$B \subseteq C$	$\Rightarrow$	$A \subseteq C$	

We will prove shortly that any subclass of a set is a set.

B.5 The Universe

The *universe* is the class of all sets, denoted **U**. It is defined by

$$\mathbf{U} = \{\, x : x = x \,\} \ .$$

From this it follows that

$$x \in \mathbf{U} \quad \Leftrightarrow \quad \mathrm{SET}(x) \ .$$

We will prove shortly that **U** is not itself a set — it is a *proper class*.

It is easy to prove now that every class is a subset of the universe:

$$\text{for every class } X, \qquad X \subseteq \mathbf{U}$$

The *Russell class*, is defined to be the class $\mathbf{RUS} = \{\, x : x \notin x \,\}$. The Russell Paradox argument shows that **RUS** is not a set. From the result above, we see that $\mathbf{RUS} \subseteq \mathbf{U}$, and so as soon as we show that a subclass of a set must be a set, it will follow that **U** is not a set.

Here is the Russell Paradox argument: note first that the definition of **RUS** is equivalent to

$$x \in \mathbf{RUS} \quad \Leftrightarrow \quad \mathrm{SET}(x) \ \wedge \ x \notin x \ .$$

Suppose that **RUS** is a set; we ask whether it is a member of itself or not. If $\mathbf{RUS} \in \mathbf{RUS}$, then it fails the condition on x in the definition displayed above, with the result that $\mathbf{RUS} \notin \mathbf{RUS}$. This is a contradiction, so we conclude that $\mathbf{RUS} \notin \mathbf{RUS}$. Since we also have the assumption SET(**RUS**), it now passes the condition on x in the definition and we have $\mathbf{RUS} \in \mathbf{RUS}$, another contradiction. Thus the assumption that **RUS** is a set leads to an inescapable contradiction; we must conclude that it is not a set.

B.6 VBG5: The Axiom of Power Sets

Let A be a class. Then the *power class* of A is defined to be the class

$$\mathcal{P}(A) = \{\, X : X \subseteq A \,\} \ ,$$

that is, the class whose members are exactly the subclasses of A which are sets — the subsets of A.

The Axiom of Power Sets states that, for every set A, there is a set W such that

$$(\forall X)(X \subseteq A \quad \Rightarrow \quad X \in W) \ ,$$

that is, there is a set W such that every subclass of A is a member of W; in other words, such that $\mathcal{P}(A) \subseteq W$.

The first thing to notice here is that, if A is any set, then every subclass of A must also be a set, since it is a member of this set W.This is the result promised in the last section.

The second thing to notice is that, again if A is a set, then its power class $\mathcal{P}(A)$ is in fact a set, since it is a subclass of the set W. We therefore call it the power *set* of A.

A corollary of all this is that the Zermelo-Fraenkel version of the Axiom of Specification holds, that is, if a is a set and $P(x)$ an expression, then

$$(\exists w \in \mathbf{U})(\forall x)(x \in w \quad \Leftrightarrow \quad x \in a \ \wedge \ P(x)) \ .$$

A couple of special power sets are worth knowing:

$$\mathcal{P}(\varnothing) = \{\varnothing\} \qquad \text{and} \qquad \mathcal{P}(\{a\}) = \{\, \varnothing, \{a\} \,\}.$$

B.7 The calculus of classes

The *complement* of a class A, usually denoted A^c, is defined by $A^c = \{\, x : x \notin A \,\}$. If A is a set, then A^c is not a set, it is a proper class (because, as we will see shortly, $A \cup A^c = \mathbf{U}$ and the union of two sets is a set).

Properties of the complement are

(i) $\varnothing^c = \mathbf{U}$ and $\mathbf{U}^c = \varnothing$.

(ii) For any class A, $(A^c)^c = A$.

(iii) For any classes A and B, $A \subseteq B \quad \Rightarrow \quad B^c \subseteq A^c$.

(iv) For any classes A and B, $A = B \quad \Rightarrow \quad A^c = B^c$.

For two classes A and B, the *class-difference* $A \setminus B$ is defined by $A \setminus B = \{\, x : x \in A \ \wedge \ x \notin B \,\}$. If A is a set, then $A \setminus B$ is also a set. When A and B are both sets, this is often called the *set-theoretic difference*.

Properties of the class-difference are:

(v) For any classes A and B, $A \setminus B \subseteq A$.

(vi) For any class A, $\quad A \setminus \varnothing = A \quad , \quad A \setminus \mathbf{U} = \varnothing \quad , \quad \varnothing \setminus A = \varnothing \quad$ and $\quad \mathbf{U} \setminus A = A^c$.

(vii) For any classes A and B, $\qquad A \subseteq B \quad \Leftrightarrow \quad A \setminus B = \varnothing \ .$

The union and intersection of two classes are defined by $A \cup B = \{\, x : x \in A \quad \text{or} \quad x \in B \,\}$ and $A \cap B = \{\, x : x \in A \quad \text{and} \quad x \in B \,\}$. There are many properties which can be listed now, of which the most important are:

(viii) $(A \cup B)^c = A^c \cap B^c \qquad$ and $\qquad (A \cap B)^c = A^c \cup B^c$

(ix) $A \cup (B \cup C) = (A \cup B) \cup C \qquad$ and $\qquad A \cap (B \cap C) = (A \cap B) \cap C$

(x) $A \cup B = B \cup A \qquad$ and $\qquad A \cap B = B \cap A$

(xi) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \qquad$ and $\qquad A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

(xii) $A \cup A = A \qquad$ and $\qquad A \cap A = A$

(xiii) $A \cup \varnothing = A \quad , \quad A \cup \mathbf{U} = \mathbf{U} \quad , \quad A \cap \varnothing = \varnothing \quad$ and $\quad A \cap \mathbf{U} = A$

xiv) $A \cup A^c = \mathbf{U} \qquad$ and $\qquad A \cap A^c = \varnothing$

(xv) $A \subseteq A \cup B \qquad$ and $\qquad A \cap B \subseteq A$

(xvi) $A \subseteq B \quad \Leftrightarrow \quad A \cup B = B \qquad$ and $\qquad A \subseteq B \quad \Leftrightarrow \quad A \cap B = A \ .$

B.8 VBG3: The Axiom of Unordered Pairs

Let a and b be classes. Then $\{a\}$ and $\{a, b\}$ are the classes defined by

$$\{a\} = \{\, x : x = a \,\} \qquad \text{and} \qquad \{a, b\} = \{\, x : x = a \ \vee \ x = b \,\} \ .$$

We note that if a is a proper class, then $\{a\}$ is empty; similarly, if a and b are proper classes, then $\{a, b\}$ is empty. Therefore these definitions are usually only used when a and b are sets. In this case they are not empty, and their defining properties are

$$x \in \{a\} \quad \Leftrightarrow \quad x = a \qquad \text{and} \qquad x \in \{a, b\} \quad \Leftrightarrow \quad x = a \ \vee \ x = b$$

In this case, $\{a\}$ is called a *singleton* and $\{a, b\}$ is called a *doubleton* or an *unordered pair*.

The Axiom of Unordered Pairs tells us that, for any sets a and b, there is a set W such that $\{a, b\} \subseteq W$. It follows from this that $\{a, b\}$ is itself a set. Also, from the definition, $\{a\} = \{a, a\}$ and so it follows that $\{a\}$ is a set also.

So for any set a, there is a set whose only member is a; for any sets a and b there is a set whose members are a and b (and no others).

From these definitions we can easily prove lots of simple facts about unordered pairs and singletons. For some examples (here a, b, c, d, x and A are assumed to be sets)

(i) $\{a, b\} = \{c, d\} \qquad$ if and only if $\qquad (a = c \wedge b = d) \quad$ or $\quad (a = d \wedge b = c) \ .$

(ii) $\{a\} = \{b\} \qquad$ if and only if $\qquad a = b \ .$

(iii) $\{a\} = \{b, c\} \qquad$ if and only if $\qquad a = b = c \ .$

(iv) $x \in A$ if and only if $\{x\} \subseteq A$.

(v) $\{a, b\} = \{\, x : x = a \vee x = b \,\}$ and $\{a\} = \{\, x : x = a \,\}$.

It is an interesting little exercise to figure out which of these statements remain true if the restriction that a, b, etc. must be sets is removed.

Before we leave singletons, note a trap for young players: the set $\{\varnothing\}$ is *not* the empty set. It has in fact one member, namely $\varnothing$, and so is a singleton set.

B.9 VBG4: Unions

There are two "kinds" of unions that turn up in mathematical discourse:

- the union of two classes $A \cup B$ (from which idea we can define the union of three classes $A \cup B \cup C$ and so on) and
- the union of a class of sets: if $\mathcal{A}$ is a class of sets this may variously be written $\bigcup_{Y \in \mathcal{A}} Y$ or $\bigcup\{\, Y : Y \in \mathcal{A} \,\}$ or, most simply, $\bigcup \mathcal{A}$. (I prefer the last and simplest notation and will use it in what follows.)

[Of course, in the world of **VBG**, the members of classes are always sets, and so any class is automatically a class of sets. When I say above that $\mathcal{A}$ is a "class of sets", it is merely a signal to the human reader that we will here actually be interested in the members of $\mathcal{A}$ as sets themselves.]

Both these kinds of union are easily defined:

$$A \cup B \quad = \quad \{\, x : x \in A \text{ or } x \in B \,\}$$

$$\bigcup \mathcal{A} \quad = \quad \{\, x : (\exists W)(x \in W \text{ and } W \in \mathcal{A} \,\}$$

Axiom VBG4 tells us that, if $\mathcal{A}$ is a *set* of sets, then so is $\bigcup \mathcal{A}$. (More precisely, it says that, if $\mathcal{A}$ is a *set* of sets, then there is a set W such that $\bigcup \mathcal{A} \subseteq W$; and from this it follows that $\bigcup \mathcal{A}$ is a set.)

These definitions can be rewritten

$$x \in A \cup B \quad \text{if and only if} \quad x \in A \text{ or } x \in B.$$

$$x \in \bigcup \mathcal{A} \quad \text{if and only if} \quad \text{there is some member } W \text{ of } \mathcal{A} \text{ such that } x \in W.$$

We can now prove various standard results, such as

$$\bigcup \varnothing = \varnothing$$

$$\bigcup \{A\} = A$$

from which, in particular,

$$\bigcup \{\varnothing\} = \varnothing.$$

Also note that, if A and B are sets, then

$$A \cup B = \bigcup \{A, B\}$$

which, with Axiom VGB3 above, tells us that, if A and B are sets, then so is $A \cup B$.

Note that $\{a, b\} = \{a\} \cup \{b\}$ and so we can extend this notation in a natural manner: $\{a, b, c\} = \{a\} \cup \{b\} \cup \{c\}$ (defined as $(\{a\} \cup \{b\}) \cup \{c\}$) and then of course we have $A \cup B \cup C = \bigcup \{A, B, C\}$ (provided A, B and C are sets) and so on.

B.10 Intersections

We define the intersection of two classes and of a class of sets in a similar way to their unions:

$$A \cap B = \{\, x : x \in A \ \wedge \ x \in B \,\}.$$

$$\bigcap \mathcal{A} = \{\, x : (\forall Y \in \mathcal{A})(x \in Y) \,\}$$

We don't need another axiom to deal with intersections. If $\mathcal{A}$ is nonempty, then its intersection is a set, even when $\mathcal{A}$ is a proper class! To see this, note that if $\mathcal{A}$ is nonempty, then it must contain at least one member, Y say, and this must be a set. It is now easy to prove that $\bigcap \mathcal{A} \subseteq Y$, and so the intersection is a set also.

On the other hand, the intersection of the empty set is not a set:

$$\bigcap \varnothing \quad = \quad \mathbf{U}$$

[*Warning*: it is not usual to use this notation when A is empty. In this case many authors consider the intersection to be undefined — as it is, in fact, in Zermelo-Fraenkel Set Theory. If you really want to use the intersection of the empty set, state explicitly what you take it to be.]

The intersection of two classes is a class (possibly a set, but not always). The intersection of two sets is always a set because of the easily proved fact that $A \cap B \subseteq A$.

We can now prove a whole swathe of such basic results as

$$A \cup B = B \cup A \quad , \quad A \cup (B \cup C) = (A \cup B) \cup C \quad , \quad A \cup A = A$$

$$A \cup \varnothing = A \quad , \quad A \subseteq B \ \Leftrightarrow \ A \cup B = B$$

$$A \cap B = B \cap A \quad , \quad A \cap (B \cap C) = (A \cap B) \cap C \quad , \quad A \cap A = A$$

$$A \cap \varnothing = \varnothing \quad , \quad A \subseteq B \ \Leftrightarrow \ A \cap B = A$$

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \quad , \quad A \cup (B \cap C) = (A \cup B) \cap (A \cup C).$$

Also, provided A and B are sets,

$$\bigcap \{A\} = A \quad , \quad \bigcap \{A, B\} = A \cap B$$

and so on; in fact now all the Calculus of Classes results listed above hold just as well for sets, except of course for those which mention complements.

B.11 Ordered pairs

Next we define an ordered pair $\langle a, b\rangle$. The crucial property of such an ordered pair is that

$$\langle a, b\rangle = \langle c, d\rangle \qquad \Leftrightarrow \qquad a = c \quad \wedge \quad b = d\ .$$

We will actually define ordered pairs twice — in two different ways. I will call the first kind a *primitive* ordered pair.

So, the *primitive ordered pair* $\langle a, b\rangle_P$ is defined by

$$\langle a, b\rangle_P = \{\{a\}, \{a, b\}\}\ .$$

As with unordered pairs, this definition makes sense when a and b are classes, but is only useful when they are both sets. Note that, if a and b are both sets, the Axiom of *unordered* pairs tells us that this *ordered* pair $\langle a, b\rangle_P$ is also a set.

Using the properties of unordered pairs listed above ((i) to (iii)), we can prove the main property of primitive ordered pairs:

B.12 Theorem

If a, b, c and d are sets, then

$$\langle a, b\rangle_P = \langle c, d\rangle_P \qquad \text{if and only if} \qquad a = c \quad \wedge \quad b = d\ .$$

We now also define (primitive) ordered triples by $\langle a, b, c\rangle_P = \langle\langle a, b\rangle_P, c\rangle_P$ and prove the crucial property, that, if a, b, c, d, e and f are all sets, then so are $\langle a, b, c\rangle_P$ and $\langle d, e, f\rangle_P$, and

$$\langle a, b, c\rangle_P = \langle d, e, f\rangle_P \qquad \text{if and only if} \qquad a = d \quad \wedge \quad b = e \quad \wedge \quad c = f\ .$$

The problem with this very primitive definition is that it only really works when the elements of the pair are sets: if A and B are proper classes, then it is not difficult to check that $\langle A, B\rangle_P = \{\varnothing\}$, irrespective of the values of A and B, and so the theorem just stated no longer holds. We will shortly have need of ordered pairs of classes, well actually an ordered triple $\langle A, G, B\rangle$ defined as $\langle\langle A, G\rangle, B\rangle$, so we now make a more general definition.

The *(ordinary) ordered pair* $\langle a, b\rangle$ is defined by

$$\langle a, b\rangle \quad = \quad \{\ \langle \varnothing, x\rangle_P\ :\ x \in a\ \}\ \cup\ \{\ \langle\{\varnothing\}, x\rangle_P\ :\ x \in b\ \}$$

Suppose that a and b are both sets. Then $\langle a, b\rangle$ is also a set. To see this, note that $\{a\}$ and $\{a, b\}$ are both subsets of $\{a, b\}$ and hence members of $\mathcal{P}(\{a, b\})$. Hence $\langle a, b\rangle_P = \{\{a\}, \{a, b\}\} \subseteq \mathcal{P}(\{a, b\})$. In particular, for any $x \in a$, $\{\varnothing, x\} \subseteq \{\varnothing\} \cup a$ and so

$$\langle \varnothing, x\rangle_P\ \subseteq\ \mathcal{P}(\{\varnothing, x\})\ \subseteq\ \mathcal{P}(\{\varnothing\} \cup a) \qquad \text{and so} \qquad \langle \varnothing, x\rangle_P\ \in\ \mathcal{PP}(\{\varnothing\} \cup a)\ .$$

Thus $\{\langle \varnothing, x\rangle_P\ :\ x\ \in\ a\}\ \subseteq\ \mathcal{PP}(\{\varnothing\} \cup a)$. In the same way $\{\langle\{\varnothing\}, x\rangle_P\ :\ y\ \in\ b\}\ \subseteq\ \mathcal{PP}(\{\{\varnothing\}\} \cup b)$ and so $\langle a, b\rangle\ = \{\langle \varnothing, x\rangle_P : x \in a\}\ \cup\ \{\langle\{\varnothing\}, x\rangle_P : y \in b\}\ \subseteq\ \mathcal{PP}(\{\varnothing\} \cup a) \cup \mathcal{PP}(\{\{\varnothing\}\} \cup b)$ is a set.

Now, first checking that $\varnothing \neq \{\varnothing\}$, we prove the main property:

B.13 Theorem

If a, b, c and d are any classes, then

$$\langle a,b\rangle = \langle c,d\rangle \qquad \text{if and only if} \qquad a=c \quad \wedge \quad b=d\ .$$

B.14 Cartesian products

The *cartesian product* $A\times B$ of two classes is defined thus:

$$A\times B \quad = \quad \{\,\langle a,b\rangle : a\in A \ \text{ and } \ b\in B\,\}$$

This is a bit informal; what it really means is

$$A\times B \quad = \quad \{\,x : x=\langle a,b\rangle \ \text{ for some } \ a\in A \ \text{ and } \ b\in B\,\}$$

or even more formally

$$A\times B \quad = \quad \{\,x : (\exists a\in A)(\exists b\in B)(x=\langle a,b\rangle)\,\}$$

As usual, we can rewrite the definition in the form

$$x\in A\times B \qquad \Leftrightarrow \qquad (\exists a\in A)(\exists b\in B)(x=\langle a,b\rangle)$$

We note that, if A and B are *sets*, then so is $A\times B$. To see this, remember that we showed that, for any a and b, $\langle a,b\rangle \subseteq \mathcal{PP}(\{\varnothing\}\cup a)\cup\mathcal{PP}(\{\{\varnothing\}\}\cup b)$. Now, for any $a\in A$, $a\subseteq\bigcup A$ and so $\{\varnothing\}\cup a\subseteq\{\varnothing\}\cup\bigcup A$. It follows that $\mathcal{PP}(\{\varnothing\}\cup a)\subseteq\mathcal{PP}(\{\varnothing\}\cup\bigcup A)$. In the same way $\mathcal{PP}(\{\{\varnothing\}\}\cup b)\subseteq\mathcal{PP}(\{\{\varnothing\}\}\cup\bigcup B)$. Thus

$$\langle a,b\rangle \ \subseteq \ \mathcal{PP}(\{\varnothing\}\cup\bigcup A)\cup\mathcal{PP}(\{\{\varnothing\}\}\cup\bigcup B)$$

and so

$$\langle a,b\rangle \ \in \ \mathcal{P}\big(\mathcal{PP}(\{\varnothing\}\cup\bigcup A)\cup\mathcal{PP}(\{\{\varnothing\}\}\cup\bigcup B)\big).$$

Since this is true for every $\langle a,b\rangle\in A\times B$, we have $A\times B \ \subseteq \ \mathcal{P}\big(\mathcal{PP}(\{\varnothing\}\cup\bigcup A)\cup\mathcal{PP}(\{\{\varnothing\}\}\cup\bigcup B)\big)$ and so $A\times B$ is a set.

We can now prove the usual properties of a cartesian product. The cartesian product is not in general commutative or associative (an interesting exercise to prove this). It does distribute over both union and intersection:

$$A\times(B\cup C)=(A\times B)\cup(A\times C) \qquad , \qquad A\times(B\cap C)=(A\times B)\cap(A\times C).$$

Some other random properties

$$A\times\varnothing=\varnothing \qquad , \qquad \varnothing\times A=\varnothing \qquad , \qquad A\times B=\varnothing \quad\Rightarrow\quad A=\varnothing \text{ or } B=\varnothing$$

$$U\subseteq A \quad \text{and} \quad V\subseteq B \quad\Rightarrow\quad U\times V\subseteq A\times B$$

$$\{a\}\times\{b\}=\{\langle a,b\rangle\}$$

and so on.

We could now go on to define cartesian products of more than two sets, for instance $A\times B\times C$ to mean $(A\times B)\times C$, but we won't because later we will have a more satisfactory way to define general cartesian products. For the time being however, the notation A^2 for $A\times A$ is useful.

B.15 Relations

First, an informal discussion leading to the mathematical definition. We now wish to define relations as mathematical objects, rather than as the relation symbols of our language. This will allow us great flexibility in creating relations of various kinds. For the time being, we will only discuss binary relations (the extension to other arities will become obvious). If R is a relation, we will denote "a stands in relation R to b" by aRb rather than $R(a, b)$. This is more in line with the notation used for such important relations as $\leq$, $<$, $\subseteq$ and so on.

Suppose now that R is a relation from a class A to a class B. Then there are three important classes here:—

A, the "domain" of R. We denote this $\mathrm{dom}(R)$.

B, the "codomain" of R. We denote this $\mathrm{cod}(R)$.

The "graph" of R, which is the set of all ordered pairs $\langle a, b\rangle$ for which aRb is true,

$$\mathrm{gr}(R) = \{\, \langle a, b\rangle : a \in A \ ,\ b \in B \text{ and } aRb \,\} \ .$$

Clearly these classes between them define the relation, $\mathrm{gr}(R)$ is a subclass of the cartesian product $A \times B$ and any subclass of $A \times B$ will define some relation from A to B. Thus we have a method of constructing relations as classes. (End of informal introduction.)

Definition A *relation* is a triple $\langle A, G, B\rangle$, where A and B are classes and G is a subclass of the cartesian product $A \times B$. In this case A is called the *domain* of R, $\mathrm{dom}(R)$; B is the *codomain* of R, $\mathrm{cod}(R)$ and G is the *graph* of R, $\mathrm{gr}(R)$. We say that R is a relation *from* A *to* B. We write aRb to mean $\langle a, b\rangle \in G$.

When we define such a relation, A and B will usually be sets, and in this case, $\mathrm{gr}(R)$ and R itself will be a set also. However it all works fine if A and B are classes.

We can now define equivalence relations:—

An *equivalence relation* on a class A is a binary relation $\equiv$ on A (that is, a relation from A to itself, as defined above) with the properties

Reflexivity	$a \equiv a$	or all $a \in A$
Symmetry	$a \equiv b \quad \Rightarrow \quad b \equiv a$	for all $a, b \in A$
Transitivity	$a \equiv b$ and $b \equiv c \quad \Rightarrow \quad a \equiv c$	for all $a, b, c \in A$

We can also define partial orders:—

A *partial order* on the class A is a binary relation $\leq$ on A with the properties

Reflexivity	$a \leq a$	for all $a \in A$
Antisymmetry	$a \leq b$ and $b \leq a \quad \Rightarrow \quad a = b$	for all $a, b \in A$

Transitivity $\quad a \leq b \quad \text{and} \quad b \leq c \quad \Rightarrow \quad a \leq c \qquad \text{for all } a, b, c \in A.$

Various other kinds of relations (preorders and full orders being a couple of important ones) can be defined the same way.

It is important to notice that now we have three distinct but related ideas, all called "relation".

(1) The relation symbols of the fully formal language. In mathematics, as defined here, there are only two of these, $\in$ and $=$.

(2) Any predicate with two free variables defines a relation, for example the predicate

$$P(a, b) \quad = \quad (\forall x)(x \in a \Rightarrow x \in b)$$

defines the subset relation.

(3) Relations defined by choosing subclasses of cartesian products, as just described.

Also, of course, where a useful relation has been defined by either methods (2) or (3), it may be given a symbol which can henceforth be used as though it was of type (1): the subset relation symbol $\subseteq$ arose this way.

B.16 Functions

We will now define a function as a special kind of relation. If f is a function $A \to B$, then we can think of $f(a) = b$ as defining a relation from A to B. If (temporarily) we write this afb to stay in line with the notation of the previous paragraph, we see that the relation f has to have a special property to qualify as a function, and arrive at the following definition.

Definition A *function* from A to B (where A and B are classes) is a relation f from A to B with the property:

For every $a \in A$ there is a unique $b \in B$ such that afb ,

in other words,

$(\forall a \in A)(\exists! b \in B)\ afb$.

This uniqueness is of course exactly what we need to use functional notation (definition by description): for any $a \in A$, we write $f(a)$ for the unique b such that afb.

(Note. We are here adopting the convention used in virtually all of mathematics except elementary calculus that, when we say that f is a function $A \to B$, we imply that it is defined on *all* of A; in other words, its domain is A, not just a subset of A.)

Note that, if A and B are sets, then any function $A \to B$ is also a set (being a subset of $A \times B$).

It is now straightforward, if tedious, to verify all the usual basic properties of functions.

We will say that two functions f and g are *equal* if they are equal as classes. This turns out to be the same as the usual meaning of equality of functions: f and g are equal if they have the same domain and codomain and $f(a) = g(a)$ for all a in their domain.

The composite of two functions and the identity function id_A on a class A are defined in the obvious ways.

Composition is associative where the composites are defined and if $f : A \to B$ then $\mathrm{id}_B \circ f = f$ and $f \circ \mathrm{id}_A = f$.

The definition of a function as a special type of relation clears up a point which sometimes causes confusion: whether there are functions $f : A \to B$ where A and or B may be empty, and if so, what they are. Perusal of the definition tells us that

- For any class B there is a unique function $\varnothing \to B$. Its graph is the empty set, so we call it the *empty function*.
- If A is nonempty, then there is no function $A \to \varnothing$.

If f is a function $A \to B$ and X is a subclass of A, we write $f[X]$ for the class of images of members of X under f, that is $f[X] = \{\, y : y \in B \ \wedge\ (\exists x \in X)(y = f(x)) \,\}$. The notation $\{\, f(x) : x \in X \,\}$ is also often used. The *range* of f is just the class $f[A]$ and is often written $\mathrm{ran} f$.

B.17 Two kinds of function and Definition by Description again

We now have two ways of defining a function.

(1) By description. If we have an expression $P(u, x)$ such that

$$(\exists! u) P(u, x) \qquad \text{for all } x$$

then we can introduce a new function symbol, defining

$$u = f(x) \qquad \text{to mean the same thing as} \qquad P(u, x).$$

(2) By specifying it as a class, as in the definition just given. If we have classes A and B and a subclass G of $A \times B$ such that, for all $a \in A$ there is a unique $b \in B$ such that $\langle a, b\rangle \in G$, then this defines a function $f = \langle A, G, B\rangle$.

These two ways of dealing with functions are equivalent, in the sense that, given any function of type (1), we can define an equivalent function of type (2) and vice versa.

However, method (2) has one outstanding advantage: if A and B are sets, then any function $A \to B$ is, as we have seen a set, and so we are able to speak of the "set of all functions $A \to B$". With method (1) this construction just doesn't make sense. Therefore, whenever new functions are introduced, it is usually tacitly understood that they are defined as sets (or classes).

B.18 Cartesian powers

For any two sets A and B we define B^A to be the set of all functions $A \to B$.

[Note the reversal of order of the symbols here!]

Also note: the set $A^\varnothing$ has exactly one member, the empty function $\varnothing \to A$. This includes the case $\varnothing^\varnothing$. On the other hand, if B is nonempty, then $\varnothing^B = \varnothing$.

We only consider the cartesian power in the case where A and B are both sets. When the definition is applied to classes it is just not useful.

B.19 Sequences

Consider a sequence $\langle a_0, a_1, a_2, \ldots\rangle$, whose elements belong to some class A. If we were to write its elements as $\langle a(0), a(1), a(2), \ldots\rangle$ instead, it would be clear that were were simply listing the values of a function $\mathbb{N} \to A$. Looking at it this way then, a sequence is nothing new; it is simply a function $\mathbb{N} \to A$ (where A is the class containing the elements of the sequence), but with a different notation which we sometimes choose to use for one reason or another.

Well ... there is something new here: we haven't got around to defining $\mathbb{N}$ yet. But when we do, we will have sequences ready made. We will also have n-tuples of course: an n-tuple $\langle a_1, a_2, \ldots, a_n\rangle$ with elements in a set A is simply a function from the subset $\{1, 2, \ldots, n\}$ of $\mathbb{N}$ to A.

Oops! Now we have *three* different definitions of an ordered pair, the ones in B.11 and B.12 above and the one just given as a 2-tuple. In the same way we have three definitions of a triple. The definitions just given (as special cases of n-tuples) are the most convenient. Unfortunately, the more primitive definitions in B.11 cannot be dispensed with — they are used to define a function, which in turn is used to define an n-tuple. Perhaps the best plan is to call both the kinds of ordered pairs and triples defined in B.11 "primitive", use them only to define relations and functions and use the more general and convenient definition of an n-tuple in all other cases. In fact, the distinction is almost never important in practice.

▶ B.11
▶ B.12

Without needing to wait for $\mathbb{N}$ to be defined, we can define families. A *family* $\{x_i\}_{i\in I}$ in a class A indexed by the set I is simply a function $x : I \to A$. (There is no reason why I cannot be a class here too, but it would be unusual for this to be so.)

In particular, if A is a set, the cartesian power A^I can be thought of as the set of all families in A indexed by I.

More generally, if $\langle A_i\rangle_{i\in I}$ is a family of sets, we can define its cartesian product to be the set of all families $\langle a_i\rangle_{i\in I}$, where $a_i \in A_i$ for all $i \in I$. This is denoted $\prod_{i\in I} A_i$ or $\prod\{A_i\}_{i\in I}$.

C Order

C.1 Definition: Various kinds of orders

An *order* on a set X is a binary relation on that set. It is usually written $\leq$, but other more exotic symbols are used from time to time. An *ordered set* is a set, X say, together with an order relation on that set: if we want to be careful, we define it as a pair $\langle X, \leq \rangle$.

It is perfectly possible to define an order relation on a *class* and to discuss ordered classes; everyting in this secion applies to ordered classes just as well as to sets.

The relevant conditions are:

(O1) *Reflexivity* For all $x \in A$, $x \leq x$.

(O2) *Transitivity* For all $x, y, z \in A$, $x \leq y$ and $y \leq z \Rightarrow x \leq z$.

(O3) *Antisymmetry* For all $x, y \in A$, $x \leq y$ and $y \leq x \Rightarrow x = y$.

(O4) *Dichotomy* For all $x, y \in A$, either $x \leq y$ or $y \leq x$.

(O5) *Well-order* Every nonempty subset of A has a least member.

A *pre-order* is a relation which satisfies Conditions O1 and O2. Pre-orders are sometimes called *quasi-orders*.

A *partial order* is a relation which satisfies Conditions O1, O2 and O3.

A *full order* is a relation which satisfies Conditions O1, O2, O3 and O4. Full orders are sometimes called *total orders* or *linear orders*. The term "linear" is best avoided in this context, since it can be confusing.

A *well-order* is a relation which satisfies all five conditions O1 – O5. We will look at this kind of order in Section E.

► E

So we have a hierarchy of order types here with stronger and stronger conditions on them as we go from pre-orders to partial orders to full orders to well-orders.

Pre-orders are not met with very often, but are occasionally useful. They are very helpful in the development of the standard number systems outlined in Appendix **A**. However we will not be considering them in the present section: here every order will be at least a partial order.

► Ap.**A**

Probably the most obvious example of a partial order which is not a full order is the subset relation $\subseteq$. This can be used as a relation on any set (or class) of sets. It is obvious that it is reflexive ($X \subseteq X$ always), antisymmetric (if $X \subseteq Y$ and $Y \subseteq X$ the $X = Y$) and transitive (if $X \subseteq Y$ and $Y \subseteq Z$ then $X \subseteq Z$). On the other hand it is not a full order except in very special circumstances (there are sets X and Y such that neither $X \subseteq Y$ nor $Y \subseteq X$). We will deal with other partial orders from time to time.

Notation: In a partially ordered class, we write

$$x \not\leq y \qquad \text{to mean} \quad \neg(x \leq y)$$

and $x < y$ to mean $(x \leq y) \ \wedge \ (x \neq y)$ — which is the same as $(x \leq y) \wedge \ (y \not\leq x)$.

The relations $x \geq y$, $x \not\geq y$ and $x > y$ are defined in the obvious ways.

It is easy to prove that, in a partially ordered class,

$$x < y \quad \Rightarrow \quad x \not\geq y \qquad \text{for all } x \text{ and } y$$

but the reverse implication does not hold in general. In fact, if the reverse implication does hold for all x and y, then the relation is a *full* order (another easy proof).

Full orders are probably the most familiar kind: the standard order relations on the natural numbers, the integers, the rationals and the reals are all full orders.

C.2 Definition of some terms

(i) Two elements a and b of a partially ordered class are *comparable* if either $a \leq b$ or $b \leq a$. Thus a fully ordered class is a partially ordered class in which all elements are comparable.

(ii) Let X be a subclass of a partially ordered class A. Then

a is a *lower bound* for X (in A) if

$$a \in A \qquad \text{and} \qquad \text{for all } x \in X, \quad a \leq x.$$

a is a *minimum* or *least element* of X if

$$a \in X \qquad \text{and} \qquad \text{for all } x \in X, \quad a \leq x,$$

that is, if it is a lower bound for X which is also a member of X.

a is a *minimal element* of X if

$$a \in X \qquad \text{and} \qquad \text{for all } x \in X, \quad x \not< a.$$

Upper bounds, *maximum* (= *greatest*) *elements* and *maximal elements* are defined in the obvious ways.

A *least upper bound* or *supremum* for X is, as its (first) name implies, a least element in the set of all upper bounds of X. The notations $\operatorname{lub} X$ and $\sup X$ are used. Similarly a *greatest lower bound* or *infimum*, denoted $\operatorname{glb} X$ or $\inf X$, is a greatest element in the set of all lower bounds of X.

(iii) A *chain* in a partially ordered class is a subclass which is fully ordered, that is, a subclass in which all elements are comparable.

(iv) A subclass X of a partially ordered class A is *initial* or an *initial segment* if

$$x \in X \quad \text{and} \quad a \leq x \qquad \Rightarrow \qquad a \in X \ .$$

For any member a of A, the set $I(a) = \{x : x < a\}$ is an initial segment (the *initial segment defined by* a), as is the class $\bar{I}(a) = \{x : x \leq a\}$. Also, A is an initial segment of itself. A partially ordered class may well have initial segments other than ones of these forms.

C.3 Suprema

The idea of a supremum should be familiar to you from calculus and analysis. Suprema and their properties will be important to us (in this chapter, but particularly in the context of ordinal numbers in Chapter **6**) so here are a number of facts worth knowing.

▶ Ch.6

(i) The supremum of a subset depends also on what it is a subset *of*. For example, let X be the set of all rational numbers x such that $x^2 < 2$. Then:

- as a subset of $\mathbb{R}$, $\sup X$ exists and is equal to $\sqrt{2}$;
- as a subset of $\mathbb{Q}$, $\sup X$ does not exist.

This suggests that the usual notation could be a bit misleading, and we should write something like, say, $\sup_{\mathbb{R}} X$ and $\sup_{\mathbb{Q}} X$. It usually doesn't matter, because we know what the "main" set is.

This same example shows that the sup does not always exist.

(ii) In any ordered set, $m = \sup X$ if and only if the following are both true:

(a) $x \leq m$ for all $x \in X$.

(b) If $a < m$, then there is some $x \in X$ such that $a < x$.

If you have a set X and element m and you want to prove that $m = \sup X$ (and there is no other obvious easy way to do it) the standard approach is to prove these two things.

(iii) In any ordered set, if the subset X has a maximum member m, then that is also its supremum. In other words, if $\max X$ exists, then $\sup X$ exists also and $\sup X = \max X$.

(iv) If a subset X happens to have a supremum and that supremum is a member of the set (that is, $\sup X$ exists and $\in X$), then it is also the maximum member of the set.

(v) Consider the empty set as a subset of some given ordered set A.

- If A has a minimum member, then $\sup \varnothing$ exists and is that minimum member,
- otherwise $\sup \varnothing$ does not exist.

For example, as a subset of $\mathbb{R}$, $\sup \varnothing$ is undefined, but as a subset of the closed interval $[3, 5]$ of reals, $\sup \varnothing = 3$.

C.4 Lemma: Initial segments of fully ordered classes

Let x and y be two members of a fully ordered class A. Then

(i) $I(x) \subseteq I(y)$ if and only if $x \leq y$ and $I(x) \subset I(y)$ if and only if $x < y$.

(ii) Either $I(x) \subseteq I(y)$ or $I(y) \subseteq I(x)$.

Proof. An easy exercise. ■

D The Natural Numbers

D.1 VBG6: The Axiom of Infinity

So far there is no axiom which allows us to create an infinite set. We now set about creating the Natural Numbers $\mathbb{N}$. We will see that most of the engineering required to manufacture this set is already available to us, with one lynchepin left to be provided by the Axiom of Infinity. Using the notation so far developed we can make this axiom look much more friendly. It tells us that there is a set W with these properties:

$$\varnothing \in W$$

$$\text{for all } x \in W, \quad x \cup \{x\} \in W \text{ also.}$$

To construct $\mathbb{N}$ we will need to define the natural number 0 and the successor x^+ of any natural number x. We define 0 to be the empty set $\varnothing$ and the successor of x by $x^+ = x \cup \{x\}$.

The set W above will contain our brand new $\mathbb{N}$, but it might be too big. We can see straight off that it contains 0 and, if it contains x then it contains x^+ also, so we are on our way. There is nothing in the axiom, however, to say that the set W does not contain a whole lot of other rubbish. So we must define $\mathbb{N}$ to be a subset of W in a crafty way. There will also be some other technical details to fix as we go. These will occupy the rest of this section.

Once we have defined $\mathbb{N}$ we want to prove Peano's Axioms in this form:

(P1) $0 \in \mathbb{N}$.

(P2) $n \in \mathbb{N}, \qquad n^+ \in \mathbb{N}$.

(P3) for all $n \in \mathbb{N}, \qquad 0 \neq n^+$.

(P4) for all $m, n \in \mathbb{N}, \qquad m^+ = n^+ \;\Rightarrow\; m = n$.

(P5) If X is a subset of $\mathbb{N}$ with the properties

$$0 \in X \qquad \text{and} \qquad (\forall x)(x \in X \;\Rightarrow\; x^+ \in X)\,,$$

then $X = \mathbb{N}$.

D.2 Definition

A set A is *inductive* if it has these properties:

$$0 \in A \qquad \text{and} \qquad (\forall x)(x \in A \;\Rightarrow\; x^+ \in A)$$

(We could define an inductive class the same way if we wished, but we don't need this generality, so why bother?)

We now define $\mathbb{N}$ by specification:

$$\mathbb{N} = \{\, n : n \text{ is a member of every inductive set} \,\}.$$

This clearly defines $\mathbb{N}$ to be a class. To see that it is a set, note that every member of $\mathbb{N}$ is a member of W (since the latter is inductive), which means that $\mathbb{N} \subseteq W$, and the axiom says that W is a set.

We define a *natural number* to be a member of $\mathbb{N}$.

D.3 Most of Peano's Axioms

We can now prove four out of the five axioms quite easily.

(P1) $\varnothing$ is a member of every inductive set by definition. Therefore it is a member of $\mathbb{N}$.

(P2) If $n \in \mathbb{N}$ then n is a member of every inductive set. But then so is n^+, and so $n^+ \in \mathbb{N}$ also.

(P3) We note that $n \in n \cup \{n\} = n^+$. Therefore $n^+ \neq \varnothing$.

(P5) If X is a subset of $\mathbb{N}$ with the given properties, then of course $X \subseteq \mathbb{N}$. But also X is inductive, so $\mathbb{N} \subseteq X$.

This leaves (P4), which is surprisingly tricky ...

D.4 Definition

A class X is *transitive* if every member is also a subset, that is,

$$(\forall x)(x \in X \quad \Rightarrow \quad x \subseteq X) \ .$$

In this context then, each such x is both a member and a subset of X.

The word *transitive* comes from the alternative (equivalent) definition: a class X is *transitive* if

$$(\forall x)(\forall y)(x \in y \text{ and } y \in X \quad \Rightarrow \quad x \in X) \ .$$

(At present we are only interested in transitive *sets*. Later, when we come to consider ordinal numbers, it will be useful to observe that the class of all ordinal numbers is transitive. But we are getting ahead of ourselves.)

D.5 Lemma

All natural numbers are transitive.

Proof. The empty set is transitive (vacuously). Since we have already proved (P5), it remains to show that, if n is a transitive natural number, then so is n^+. Suppose then that n is such a number and that $x \in n^+$; we want to show that $x \subseteq n^+$. We have $x \in n \cup \{n\}$, so either $x \in n$ or $x \in \{n\}$. If $x \in n$ then $x \subseteq n$ (by the inductive hypothesis) and then, since $n \subseteq n^+$ we have $x \subseteq n^+$. On the other hand, if $x \in \{n\}$ then $x = n$ and again $x \subseteq n^+$. ■

Proof of P4. First note a couple of facts:

(1) If n is a natural number, then $n \in n^+$.

(2) If m and n are natural numbers and $m \in n^+$ then $m \subseteq n$.

Here (1) follows immediately from $n^+ = n \cup \{n\}$. For (2), we have $m \in n \cup \{n\}$ so either $m \in n$ or $m = n$; if $m = n$ we are done and if $m \in n$ we have $m \subseteq n$ by transitivity.

Now to prove P4. Given $m^+ = n^+$, we have $m \in m^+$ by (1), so $m \in n^+$ and then $m \subseteq n$ by (2). Similarly $n \subseteq m$. Done. ■

One would expect there to be a theorem stating that no set can be a member of itself. This is in fact so, the result following from the Axiom of Foundation, and we will prove it as Theorem G.3.

▶ G.3

It is nice however to know that this esoteric axiom is not needed to prove such a basic fact for the natural numbers; it follows from what we have just proved:

D.6 Lemma

No natural number is a member of itself (that is, if $n \in \mathbb{N}$ then $n \notin n$).

Proof. The empty set has no members, so it is not a member of itself. It now suffices to show that if n is a natural number which is not a member of itself, then so is n^+. Suppose then that n is such a number, $n \notin n$. We want to show that $n^+ \notin n^+$.

Proof is by contradiction: suppose that $n^+ \in n^+$, that is, $n^+ \in n \cup \{n\}$. Then either $n^+ \in n$ or $n^+ \in \{n\}$. But if $n^+ \in n$ then $n^+ \subseteq n$, since n is transitive. If, on the other hand, $n^+ \in \{n\}$, then $n^+ = n$. So in either case $n^+ \subseteq n$. But $n \in n^+$, so then $n \in n$, a contradiction. ■

D.7 Fun and games

This definition of $\mathbb{N}$ has some very pretty properties. We can list the first few natural numbers as

$$0 = \varnothing \quad , \quad 1 = \varnothing^+ \quad , \quad 2 = \varnothing^{++} \quad , \quad 3 = \varnothing^{+++} \quad , \quad \text{etc.}$$

Expanding this out gives

$$0 = \varnothing \quad , \quad 1 = \{\varnothing\} \quad , \quad 2 = \{\varnothing, \{\varnothing\}\} \quad , \quad 3 = \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\} \quad , \quad \ldots$$

and another interesting way of looking at all this is

$$0 = \varnothing \quad , \quad 1 = \{0\} \quad , \quad 2 = \{0, 1\} \quad , \quad 3 = \{0, 1, 2\} \quad , \quad 4 = \{0, 1, 2, 3\} \quad , \quad \ldots$$

and in general, $n = \{0, 1, 2, \ldots, n-1\}$.

Also (assuming that we have sorted out what *counting* means), each n has exactly n members. Indeed, we can define counting by saying that a set has n members if it can be placed in one-to-one correspondence with (the set) n, and we can define a *finite* set as one that can be so counted (by some n).

D.8 Definition by induction

The Induction axiom P5 allows us to prove things about the natural numbers by induction. We still have to see how to define things by induction. In the simplest case we want to be able to define a function $f : \mathbb{N} \to B$, where B is some class, by specifying $f(0) \in B$ and $f(n^+)$ in terms of n and $f(n)$.

D.9 Theorem: Definition by Induction, no parameters

Let B be a class, b a member of B and $h : \mathbb{N} \times B \to B$ be a function. Then there is a unique function $f : \mathbb{N} \to B$ satisfying

$$f(0) = b$$

and

$$f(n^+) = h(n, f(n)) \qquad \text{for all } n \in \mathbb{N}.$$

More generally, there may be other variables involved and we will want to define $f(a_1, a_2, \ldots, a_k, n)$. In this case we can of course consider the n-tuple $\langle a_1, a_2, \ldots, a_k \rangle$ to be a single variable, and so our general problem boils down to defining a function $f : A \times \mathbb{N} \to B$ by specifying $f(a, 0)$ for every $a \in A$ and specifying $f(a, n^+)$ in terms of a, n and $f(a, n)$ for all $a \in A$ and $n \in \mathbb{N}$.

Thus we have

D.10 Theorem: Definition by Induction, with parameters

Let A and B be classes, $g : A \to B$ and $h : A \times \mathbb{N} \times B \to B$ be functions. Then there is a unique function $f : A \times \mathbb{N} \to B$ satisfying

$$\begin{aligned} f(a, 0) &= g(a) && \text{for all } a \in A \quad \text{and} \\ f(a, n^+) &= h(a, n, f(a, n)) && \text{for all } a \in A \text{ and } n \in \mathbb{N}. \end{aligned} \tag{–1}$$

Proof. The proof that f is unique is a simple application of proof by induction and will be left as an exercise. The fact that such an f exists is what will be proved here.

For the purposes of this proof, let us write I_n for the subset $\{0, 1, \ldots, n\}$ of $\mathbb{N}$ and define a *starter* to be a function $\sigma : A \times I_n \to B$ for some $n \in \mathbb{N}$ satisfying

$$\begin{aligned} \sigma(a, 0) &= g(a) && \text{for all } a \in A \quad \text{and} \\ \sigma(a, x^+) &= h(a, x, \sigma(a, x)) && \text{for all } a \in A \text{ and } x < n\,. \end{aligned} \tag{–2}$$

An easy induction shows that, if there exists a starter $A \times I_n \to B$ then it is unique. This allows us to use the following notation: if $n \in \mathbb{N}$ and a starter $A \times I_n \to B$ exists, then call it σ_n.

Now we show that, for every $n \in \mathbb{N}$, there does exist a starter $A \times I_n \to B$. The proof is by induction again. Defining $\sigma_0 : A \times I_0 \to B$ by $\sigma_0(a, 0) = g(a)$, it clearly satisfies (–1)

and so is a starter. Now suppose that there exists a starter $\sigma : A \times I_n \to B$ and define $\sigma' : A \times n^+ \to B$ by

$$\begin{aligned} \sigma'(a,x) &= \sigma_n(a,x) && \text{for all } x \leq n \quad \text{and} \\ \sigma'(a,n^+) &= h(a,n,\sigma(a,n))\,. \end{aligned} \tag{–3}$$

Then σ' satisfies (–2) for all $x \leq n$ since σ is a starter, and for $x = n^+$ by its definition. Thus σ' is the required starter $A \times I_{n^+} \to B$.

The existence and uniqueness of starters just proven allow us to make the following definition: for any $n \in \mathbb{N}$, σ_n is *the* starter $A \times I_n \to B$. Consequently, equations (–3) above actually define σ_{n^+}, so we see that, for any $n \in \mathbb{N}$,

$$\sigma_n(a,n) = \sigma_{n+}(a,n) \tag{–4}$$

Now we define $f : A \times \mathbb{N} \to B$ by

$$f(a,n) = \sigma_n(a,n) \qquad \text{for all } a \in A \text{ and } n \in \mathbb{N}\ .$$

It remains to show that f, with this definition, satisfies equations (–1). Now $f(a,0) = \sigma_0(a,0) = g(a,0)$, since σ_0 is a starter. Also, for any $n \in \mathbb{N}$,

$$\begin{aligned} f(a,n^+) &= \sigma_{n^+}(a,n^+) && \text{by definition of } f \\ &= h(a,n,\sigma_{n^+}(a,n)) && \text{by Equation (–2) for } \sigma_{n^+} \\ &= h(a,n,\sigma_n(a,n)) && \text{by Equation (–4)} \\ &= h(a,n,f(a,n)) && \text{by definition of } f \text{ again}\ . \end{aligned}$$

■

D.11 Where to now?

Now we can define addition and multiplication inductively, using the definitions in the Chapter 4 but the techniques of this chapter. We have much more powerful machinery at our disposal now, so the development is much easier. Also, of course, we are now *allowed* to talk about sets and classes, so we can say much more.

We can also define the order on $\mathbb{N}$ as in Chapter 4, or else we can ignore addition and define it directly: take a hint from the fact noted above that $n = (1, 2, \ldots, n-1)$, and define $a \leq b$ to mean $a \in b \ \vee \ a = b$. It is an interesting exercise to prove the usual order properties from this basis.

More ambitiously, we develop all the well-known algebraic and order properties of $\mathbb{N}$. Having done this, we construct in turn The Integers $\mathbb{Z}$, The Rationals $\mathbb{Q}$, The Reals $\mathbb{R}$ and The Complex Numbers $\mathbb{C}$ and prove their defining properties.

These ideas will be taken up in Appendix **A**; meanwhile we look at the idea of strong induction in a more general context. ▶ Ch.A

E Well-ordering

E.1 Notes

Well-order was defined in Section C.1. (The various terms that were defined in that section will be used here.) With regard to that definition, note that:—

(i) Properties O3 and O5 together imply the others, so in order to verify that a relation is a well-order, it is sufficient to verify these two properties. (Proof as an exercise).

(ii) When we come to discuss the Axiom of Choice, we will see that that axiom is equivalent to the *Well-Ordering Principle*: Every set can be well-ordered.

(iii) It is a straightforward exercise to prove that $\mathbb{N}$ is well-ordered.

E.2 Lemma: Initial segments of well-ordered classes

(i) Let w be a member of a fully ordered class W. Then w is the least member of $W \setminus I(w)$.

(ii) An initial segment of a well-ordered class W is either W itself or else of the form $I(w)$ for some $w \in W$.

Proof. Another easy exercise. ■

E.3 Lemma: Strong induction

There are two ways of looking at strong induction:

(i) Let W be a well-ordered class and E be a subclass of W such that

$$I(x) \subseteq E \quad \Rightarrow \quad x \in E \qquad \text{for all } x \in W.$$

Then $E = W$.

(ii) Let W be a well-ordered class and $P(x)$ be an expression such that

$$(\forall x < a)P(x) \quad \Rightarrow \quad P(a) \qquad \text{for all } a \in W.$$

Then $P(a)$ is true for all $a \in W$.

Proof. (i) Suppose $E \neq W$. Then $W \setminus E$ is nonempty and so has a least element, x say. Then $I(x) \subseteq E$ but $x \notin E$, contradicting the assumption.

(ii) is an immediate corollary of (i).

■

E.4 Definition

Let A and B be two partially ordered classes.

(i) A function $f : A \to B$ is *order-preserving* if, for any $a_1, a_2 \in A$

$$a_1 \leq a_2 \quad \Rightarrow \quad f(a_1) \leq f(a_2) \ .$$

(ii) An *order-isomorphism* is a bijection $\varphi : A \to B$ such that, for any $a_1, a_2 \in A$

$$a_1 \leq a_2 \quad \Leftrightarrow \quad \varphi(a_1) \leq \varphi(a_2) \ .$$

An alternative description of an order isomorphism is as a bijection such that both it and its inverse are order-preserving. Note that, if both A and B are known to be fully ordered, then either $\Rightarrow$ or $\Leftarrow$ is enough in (ii) above. Putting that another way, if both A and B are known to be fully ordered, then an order-preserving bijection must be an order-isomorphism.

E.5 Lemma

(i) Suppose that X and Y are fully ordered classes and $f : X \to Y$ is an order-preserving bijection. Then $f[I(x)] = I(f(x))$ for all $x \in X$.

(ii) Let W be a well-ordered class and $f : W \to W$ be one-to-one and order-preserving. Then $x \leq f(x)$ for all $x \in W$.

(iii) Let W be a well-ordered class. Then there is only one order isomorphism $W \to W$, namely the identity function.

(iv) Let W be a well-ordered class. Then W is not order isomorphic to $I(x)$ for any $x \in W$.

Proof. (i) Suppose $y \in f[I(x)]$. Then there is $u \in I(x)$ such that $y = f(u)$. Then $u < x$ and, since f is one-to-one, $f(u) < f(x)$, i.e. $y \in I(f(x))$.

Conversely, suppose that $y \in I(f(x))$. Since f is onto, there is $u \in X$ such that $y = f(u)$. Then $u \in I(x)$, for otherwise (X being fully ordered) $x \leq u$, in which case we would have $y = f(u) \geq f(x)$, contradicting $y \in I(f(x))$.

(ii) Observe first that the conditions on f tell us that, if $x < y$ in W, then $f(x) < f(y)$ (f is *strict order-preserving*). Let E be the class of all members x of W such that $x \leq f(x)$; we want to prove that $E = W$. Suppose not; then $E \setminus E$ is nonempty and so has a least member, w say. Then $f(w) < w$, from which we deduce two things, first that $f(w) \in E$ and second that $f(f(w)) < f(w)$. But these contradict one another.

(iii) and **(iv)** are immediate corollaries of (i) and (ii).

■

E.6 Theorem: Trichotomy theorem for well-ordered classes

Let A and B be two well-ordered classes. Then exactly one of the following holds

(i) A and B are order-isomorphic.

(ii) There is some $y \in B$ such that A is order-isomorphic to $I(y)$.

(iii) There is some $x \in A$ such that B is order-isomorphic to $I(x)$.

Furthermore,

in Case (i), the order-isomorphism $A \to B$ is unique,

in Case (ii), the element y of B is unique as is the order-isomorphism $A \to I(y)$ and

in Case (iii), the element x of A is unique as is the order-isomorphism $B \to I(x)$.

Proof. This is all a corollary of the previous lemma. ■

E.7 Corollary

From this theorem we can say that, if A and B are two well-ordered classes, then either A is order-isomorphic to an initial segment of B or B is order-isomorphic to an initial segment of A. Moreover, if both, then A and B are themselves order-isomorphic.

E.8 Definition by induction over a well-ordered set

▶ D.10

This theorem is an extension of the idea of definition by induction over $\mathbb{N}$ (Theorem D.10). Since we are now dealing with a well-ordered set, we use a definition technique which corresponds to strong induction.

E.9 Theorem

Let A and B be classes, W a well-ordered class and h be a function $h : A \times W \times \mathcal{I} \to B$, where $\mathcal{I}$ is the class of all functions $A \times I(w) \to B$ for all $w \in W$. Then there is a unique function $f : A \times W \to B$ such that

$$f(a, x) \;=\; h(a, x, f|_{A \times I(x)}) \qquad\qquad \text{for all } a \in A \text{ and } x \in W.$$

Proof. For the purposes of this proof, define a *starter* to be such a function, but defined on an initial segment $\bar{I}(w)$ of W, that is, a function $\sigma : A \times \bar{I}(w) \to B$, where w is any member of W and

$$\sigma(a, x) \;=\; h(a, x, \sigma|_{A \times I(x)}) \qquad\qquad \text{for all } a \in A \text{ and } x \in \bar{I}(w). \tag{–1}$$

We show first that, for any $w \in W$,

$$\text{if there is a starter } \sigma : A \times \bar{I}(w) \to B, \text{ then it is unique.} \tag{–2}$$

Suppose then that we have two starters $\sigma, \sigma' : A \times \bar{I}(w) \to B$ and $\sigma \neq \sigma'$. Then there is some $a \in A$ and $x \in \bar{I}(w)$ such that $\sigma(a, x) \neq \sigma'(a, x)$; using well-order, we may suppose that x is in fact the least such. Then we have $\sigma(a, z) = \sigma'(a, z)$ for all $z < x$; in other words, $\sigma|_{A \times I(x)} = \sigma'|_{A \times I(x)}$. But then

$$\sigma(a, x) \;=\; h(a, x, \sigma|_{A \times I(x)}) \;=\; h(a, x, \sigma'|_{A \times I(x)}) \;= \sigma'(a, x)\,,$$

contradicting the choice of x and so proving (–2).

Now let S be the subclass of all $w \in W$ such that there does exist a starter $A \times \bar{I}(w) \to B$. From the uniqueness just proved we are justified in using the notation: if $w \in S$, denote the unique starter $A \times \bar{I}(w) \to B$ by σ_w.

Now suppose that $w \in S$ and $v \leq w$, and consider the function $\sigma' = \sigma_w|_{A \times \bar{I}(w)}$. Noting that, for any $x \leq v$ we have $\sigma'(a,x) = \sigma_w(a,x)$, so that $\sigma'|_{A \times I(x)} = \sigma_w|_{A \times I(x)}$, we see that, for any $x \leq v$,

$$\sigma'(a,x) \;=\; \sigma(a,x) \;=\; h(a,x,\sigma|_{A \times I(x)}) \;=\; h(a,x,\sigma'|_{A \times I(x)})$$

and so σ' is a starter and therefore the starter $A \times \bar{I}(v) \to B$, that is $\sigma' = \sigma_v$. From this we see several things. Firstly, if $w \in W$ and $v \leq w$, then $v \in S$ also, that is, S is an initial segment of W. Secondly, if $v \leq w \in S$,

$$\sigma_v \;= \sigma_w|_{A \times \bar{I}(v)}\,.$$

And finally, again if $v \leq w \in S$,

$$\sigma_v(a,x) \;=\; \sigma_w(a,x) \qquad\qquad \text{for all } a \in A \text{ and } x \leq v.$$

Now let us define the function $f : A \times S \to B$ by

$$f(a,w) \;= \sigma_w(a,w)$$

We note that, for $v \leq w$,

$$f(a,v) \;=\; \sigma_v(a,v) \;=\; \sigma_w(a,v)$$

which tells us that

$$f|_{A \times \bar{I}(w)} \;=\; \sigma_w\,.$$

Noting that $I(w) \subseteq \bar{I}(w)$, we have $\sigma_w|_{A \times I(w)} = f|_{A \times I(w)}$ so,

$$\begin{aligned} f(a,w) \;&=\; \sigma_w(a,w) && \text{by definition of } f \\ &=\; h(a,w,\sigma_w|_{A \times I(w)}) && \text{since } \sigma_w \text{ is a starter} \\ &=\; h(a,w,f|_{A \times I(w)}) && \text{just noted} \end{aligned}$$

so f satisfies the main equation in the form

$$f(a,w) \;=\; h(a,w,f|_{A \times I(w)}) \qquad\qquad \text{for all } a \in A \text{ and } w \in S.$$

It remains only to show that $S = W$. Suppose not. Then $W \setminus S \neq \varnothing$ so, using well-ordering, let w be the least member of $W \setminus S$. Then there is a starter $\sigma_x : A \times \bar{I}(x) \to B$ for all $x < w$, which means that $I(w) \subseteq S$. Define $\sigma : A \times \bar{I}(w) \to B$ by

$$\sigma(a,x) \;=\; \begin{cases} f(a,x) & \text{if } x < w, \\ h(a,w,f|_{A \times I(w)}) & \text{if } x = w. \end{cases}$$

But this σ is also a starter. To see this we check (–1). Firstly, for $x < w$, $\sigma(a,x) = f(a,x)$ which tells us that $\sigma|_{A \times I(w)} = f|_{A \times I(w)}$, and so

$$\sigma(a,w) = h(a,w,f|_{A \times I(w)}) = h(a,w,\sigma|_{A \times I(w)})\,.$$

Noting also that, for any $x < w$, $\sigma(a,x) = f(a,x) = \sigma_x(a,x)$, so $\sigma|_{A\times I(x)} = \sigma_x|_{A\times I(x)}$,

$$\begin{aligned}
\sigma(a,x) &= f(a,x) && \text{by definition of } \sigma \\
&= \sigma_x(a,x) && \text{by definition of } f \\
&= h(a,x,\sigma_x|_{A\times I(x)}) && \text{since } \sigma_x \text{ is a starter} \\
&= h(a,x,\sigma|_{A\times I(x)}) && \text{just noted.}
\end{aligned}$$

This proves that σ is a starter $A\times\bar{I}(w) \to B$, contradicting the choice of w and thus proving that $S = W$, as required. ■

F The Axiom of Choice

F.1 The axiom

The Axiom of Choice, as usually stated, is

(AC) Let $\mathcal{A}$ be a set of nonempty sets. Then there is a function $f : \mathcal{A} \to \bigcup \mathcal{A}$ such that $f(A) \in A$ for all $A \in \mathcal{A}$.

(Think of f as "choosing" one element of each set in $\mathcal{A}$.)

F.2 Discussion

The Axiom of Choice is *not* one of the axioms of von Neumann-Bernays-Gödel Set Theory (**VBG**). I will call Set Theory with the Axiom of Choice added **VBG**+AC.

The Axiom of Choice has been shown to be *independent* of **VBG**, that is,

(1) AC is consistent with Set Theory: if **VBG** is consistent, then so is **VBG**+AC.

(2) ¬AC is also consistent with Set Theory: if **VBG** is consistent, then so is **VBG**+¬AC.

The Axiom of Choice has a special status: some mathematicians are happy to use it, others regard it with deep suspicion. If you use an argument which involves AC then it is a good idea to say so. Moreover, if it is possible to find an alternative argument which does not involve AC, this is generally considered to be a good thing.

Given the controversial nature of AC, the first consistency result above is reassuring. Even if AC does not have the acceptance of the other axioms, at least its use cannot involve any more risk of a self contradiction that mathematics without it. The second consistency result shows that the search for a proof of AC from the other axioms is futile (unless, of course, you suspect that **VBG** itself is inconsistent, and that this might be a good way to prove it!).

Because of the status of the Axiom of Choice, I will decorate results in these notes which depend upon it with a superscript thus $^{\boxed{\text{AC}}}$.

F.3 Equivalent "axioms"

There are several very useful statements which are equivalent to the Axiom of Choice (that is, they are formally equivalent to AC in **VBG**.) Thus anything which requires any of these "axioms" can be taken as involving the use of AC.

Three other ways the Axiom of Choice is sometimes stated (equivalent of course) are:

$(\text{AC}_1)^{\boxed{\text{AC}}}$ Let A be a set. Then there is a function $c : \mathcal{P}(A) \setminus \{\varnothing\} \to A$ such that $c(X) \in X$ for every nonempty subset X of A.

(The function c here is called a *choice function* for A.)

$(\text{AC}_2)^{\boxed{\text{AC}}}$ The cartesian product of any family of nonempty sets is nonempty.

(AC_3) AC Let $\mathcal{A}$ be a set of pairwise disjoint nonempty sets. Then there is a set C which contains exactly one element from every member of $\mathcal{A}$.

(The set C here is called a *choice set* for A.)

The last form is the one which is most easily expressed in the formal language (without need for preliminary definitions). It can be stated:

$$\big((\forall u)(u \in a \;\Rightarrow\; u \neq \varnothing) \quad \wedge \quad (\forall u)(\forall v)(u \in a \wedge v \in a \wedge (\exists x)(x \in u \wedge x \in v) \quad \Rightarrow \quad u = v) \big)$$

$$\Rightarrow \quad (\exists c)\big[\quad (\forall u)(u \in a \quad \Rightarrow \quad (\exists x)(x \in c \;\wedge\; x \in u))$$

$$\wedge \quad (\forall x)(\forall y)(\, x \in c \wedge y \in c \wedge (\exists u)(x \in u \wedge y \in u \wedge u \in a) \quad \Rightarrow \quad x = y \,)\,\big].$$

These four formulations of the Axiom of Choice are easily proved equivalent. It is more difficult to prove the following three equivalent. They are, however, very useful results, so the proofs will be given here.

Maximal Principle AC Every partially ordered set has a maximal chain.

Zorn's Lemma AC Let P be a nonempty partially ordered set in which every chain has an upper bound. Then P has a maximal element.

The Well-Ordering Principle AC For every set there is a well-order.

(This is often expressed: "Every set can be well-ordered". This is a tad dangerous, if you interpret it to mean that you can explicitly *specify* or *construct* a particular well-order for the set. For many sets you cannot.)

In order to prove these are equivalent to the Axiom of Choice, we start with another lemma, also equivalent, but not particularly interesting in its own right. (Its proof does however contain most of the hard work here.)

F.4 Lemma AC

Let X be a set and $\mathbb{S}$ be a collection of subsets of X (which we will consider to be partially ordered by inclusion), with the properties:

(1) $\varnothing \in \mathbb{S}$

(2) If $V \in \mathbb{S}$ and $U \subseteq V$, then $U \in \mathbb{S}$ also.

(3) If $\mathbb{C}$ is a chain in $\mathbb{S}$, then $\bigcup \mathbb{C} \in \mathbb{S}$.

Then $\mathbb{S}$ has a maximal member.

Proof that AC $\Rightarrow$ this lemma. By the Axiom of Choice, there is a choice function $c : \mathcal{P}(X) \setminus \{\varnothing\} \to X$ (so that, for every nonempty subset U of X, $c(U) \in U$).

For each member S of $\mathbb{S}$, define a set $\bar{S}$ by

$$\bar{S} = \{\, x \in X : S \cup \{x\} \in \mathbb{S} \,\}$$

Obviously $S \subseteq \bar{S}$ for all such S.

Suppose that $S \neq \bar{S}$. Then $S \subset \bar{S}$, so there is some $x \notin S$ such that $S \cup \{x\} \in \bar{S}$. This means that S is not maximal in $\mathbb{S}$. Conversely, suppose that S is not maximal. Then there is some $T \in \mathbb{S}$ such that $S \subset T$. Then there is some $x \in T \setminus S$ and then $S \cup \{x\} \subseteq T$, so $S \cup \{x\} \in \mathbb{S}$ by Condition (2). But then $x \in \bar{S}$. Since $x \notin S$, $S \neq \bar{S}$. We have just proved that $S \neq \bar{S}$ if and only if S is not maximal. In other words, S is maximal if and only if $S = \bar{S}$. Our task becomes: show that S contains a member S such that $S = \bar{S}$.

For each set $S \in \mathbb{S}$, define a set S^* by

$$S^* = \begin{cases} S \cup c(\bar{S} \setminus S) & \text{if } \bar{S} \setminus S \neq \varnothing; \\ S & \text{otherwise.} \end{cases}$$

We note that, if $S \in \mathbb{S}$ then $S^* \in \mathbb{S}$ also. Moreover, if S is maximal, $\bar{S} \setminus S = \varnothing$ and so $S^* = S$. If S is not maximal, then $S \subset S^*$ and $S^* \setminus S$ is a singleton. Our task becomes: show that $\mathbb{S}$ contains a member S such that $S = S^*$.

We make a definition for the purposes of this proof: a subcollection $\mathbb{T}$ of $\mathbb{S}$ is a *tower* if

(i) $\varnothing \in \mathbb{T}$

(ii) $T \in \mathbb{T} \quad \Rightarrow \quad T^* \in \mathbb{T}$

(iii) if $\mathbb{C}$ is a chain in $\mathbb{T}$, then $\bigcup \mathbb{C} \in \mathbb{T}$.

There is at least one tower, namely $\mathbb{S}$ itself. Also the intersection of any collection of towers is a tower. Therefore there is a "smallest" tower, $\mathbb{T}_0$ say — define $\mathbb{T}_0$ to be the intersection of all towers and then it is a tower which is a subset of any other tower.

We will show that $\mathbb{T}_0$ is a chain; this will prove the lemma, for then, writing W for $\bigcup \mathbb{T}_0$, we have $W \in \mathbb{T}_0$ by (iii). Then from (ii), $W^* = W$. Also, $\mathbb{T}_0$ is a chain in $\mathbb{S}$, so $W \in \mathbb{S}$. To this end, we will say that a member C of $\mathbb{T}_0$ is *central* if it is comparable with every member of $\mathbb{T}_0$, that is

$$\text{for all } T \in \mathbb{T}_0, \qquad \text{either } \ T \subseteq C \text{ or } C \subseteq T.$$

It is now enough to show that every member of $\mathbb{T}_0$ is central and, to do this, it is sufficient to show that the set of all central members of $\mathbb{T}_0$ is a tower; this is now our task.

Firstly, $\varnothing$ is obviously central. Now let $\mathbb{K}$ be a chain of central elements of $\mathbb{T}_0$; we show that $\bigcup \mathbb{K}$ is central. Let T be any member of $\mathbb{T}_0$; we want to show that T and $\bigcup \mathbb{K}$ are comparable. If every member of $\mathbb{K}$ is a subset of T then so is its union. Otherwise there is some $K \in \mathbb{K}$ such that $K \nsubseteq T$. But then since K is central, $T \subseteq K$ and then $T \subseteq \bigcup \mathbb{K}$.

It remains to show that if C is central then so is C^*. From now on then suppose that C is a fixed central member of $\mathbb{T}_0$. We consider the collection $\mathbb{U}$ of all sets U in $\mathbb{T}_0$ such that either $U \subseteq C$ or $C^* \subseteq U$. We will show that $\mathbb{U}$ is a tower. This proves the lemma, for then $\mathbb{U} = \mathbb{T}_0$ (by definition of $\mathbb{T}_0$) and so, for every member T of $\mathbb{T}_0$ we have $T \subseteq C$ or $C^* \subseteq T$; since $C \subseteq C^*$, this makes C^* central.

It remains to show that $\mathbb{U}$ is a tower. Clearly $\varnothing \in \mathbb{U}$. Now let $\mathbb{K}$ be a chain in $\mathbb{U}$. Then either every member of $\mathbb{K}$ is a subset of C, in which case so is $\bigcup \mathbb{K}$, or else $\mathbb{K}$ has a member, K say, which is not a subset of C, and then $C^* \subseteq \mathbb{K}$ so $C^* \subseteq \bigcup \mathbb{K}$.

It now remains to show that, if $U \in \mathbb{U}$ then $U^* \in \mathbb{U}$. If $C^* \subseteq U$ then $C^* \subseteq U^*$ and we are done. If $U = C$ then $U^* = C^*$ and again we are done. Finally suppose $U \subset C$. Since C is central, either $U^* \subseteq C$ or $C \subset U^*$. If $U^* \subseteq C$ then once again we are done, and $C \subset U^*$ is impossible, for then we would have $U \subset C \subset U^*$ and $U^* \setminus U$ is a singleton. ■

Proof that Lemma E4 ⇒ the Maximum Principle. Let X be a partially ordered set and let $\mathbb{S}$ be the collection of all chains in X. Now $\varnothing$ is a chain in X, so $\varnothing \in \mathbb{S}$. If V is a chain in X and $U \subseteq V$ then U is a chain also. Now suppose that $\mathbb{C}$ is a chain in $\mathbb{S}$ (under $\subseteq$ as usual); we prove that $\bigcup \mathbb{C} \in \mathbb{S}$, that is, its union is a chain in X.

Let $a, b \in \bigcup \mathbb{C}$. Then there are sets $A, B \in \mathbb{C}$ such that $a \in A$ and $b \in B$. Since $\mathbb{C}$ is a chain, A and B are comparable; suppose without loss of generality that $A \subseteq B$. Then $a, b \in B$. Since B is a chain, a and b are comparable. ■

Proof that the Maximum Principle ⇒ Zorn's Lemma. By the Maximum Principle, P has a maximal chain, C say. By hypothesis, C has an upper bound, m say. We show that m is a maximal member of P. Suppose not. Then there is $p \in P$ such that $m < p$. It follows that $C \cup \{p\}$ is also a chain, contradicting the maximality of C. ■

Proof that Zorn's Lemma ⇒ the Well-Ordering Principle. (I will only give an outline of the proof here. The main structure of the proof is a classic example of how Zorn's Lemma is used. The details are straightforward but a bit tedious, and can safely be left as an exercise.)

Let X be any set; we show that it can be well-ordered. Let $\mathbb{W}$ be the collection of all well-ordered subsets of X, that is, all pairs of the form $\langle W, \leq \rangle$, where W is a subset of X and $\leq$ is a well-order of W. We now order $\mathbb{W}$ in the obvious (I think!) way:

$$\langle W_1, \leq_1 \rangle \ \leq \ \langle W_2, \leq_2 \rangle \qquad \text{if and only if} \qquad \langle W_1, \leq_1 \rangle \text{ is an initial segment of } \langle W_2, \leq_2 \rangle,$$

that is, if

$$W_1 \subseteq W_2 \ ,$$

$$\leq_1 \text{ is the restriction of } \leq_2 \text{ to } W_1$$

and

$$u \leq v \text{ in } W_2 \ \text{ and } \ v \in W_1 \quad \Rightarrow \quad u \in W_1 \ .$$

We now use Zorn's Lemma to show that $\mathbb{W}$ has a maximal member. To do this, we suppose that $\mathbb{C}$ is a chain in $\mathbb{W}$ and construct an upper bound for $\mathbb{C}$. This upper bound is the union of all the sets in $\mathbb{C}$ with the "obvious" order. More precisely, it is the pair $\langle M, \leq_0 \rangle$ defined by

$$M = \bigcup \{\, W : \langle W, \leq \rangle \in \mathbb{C} \,\} \qquad \text{(the union of all the sets "in" the chain)}$$

and, for any $x, y \in M$,

$$x \leq_0 y \qquad \text{if and only if} \qquad \text{there is some } \langle W, \leq \rangle \in \mathbb{C} \ \text{ such that } x, y \in W \text{ and } x \leq y.$$

The "straightforward but tedious" details that need checking here are that $\langle M, \leq_0 \rangle$ is indeed a member of W, that is, M is a subset of X and $\leq_0$ is a well-order of M, and that it is an upper bound for $\mathbb{C}$.

Having established that $\mathbb{W}$ does have a maximal member, $\langle W, \leq \rangle$ say, it is enough to show that then $W = X$. Suppose not. Then there is some $x \in X$ such that $x \notin W$. But we can extend the well-order of W to $W \cup \{x\}$ by making x a new greatest element — more precisely, define an order $\leq'$ on $W \cup \{x\}$ by

$$u \leq' v \text{ in } W \cup \{x\} \quad \text{if and only if} \quad \text{either (i) } u, v \in W \text{ and } u \leq v \text{ in } W \text{ or (ii) } v = x.$$

It is now easy to check that $\leq'$ well-orders $W \cup \{x\}$ and that $\langle W, \leq \rangle$ is an initial segment of $\langle W \cup \{x\}, \leq' \rangle$, contradicting the maximality of $\langle W, \leq \rangle$. ■

Proof that the Well-Ordering Principle ⇒ the Axiom of Choice. Let X be any set. We show that there is a choice function $c : \mathcal{P}(X) \setminus \{\varnothing\} \to X$. There is a well-order, $\leq$ say of X. Now define the function c by defining its graph:

$$\mathrm{gr}(c) = \{\, \langle A, \text{least member of } A \rangle : A \in \mathcal{P}(X) \setminus \{\varnothing\} \,\}$$

■

G The last two axioms

G.1 VBG8: The Axiom of Foundation

There are several equivalent formulations of the Axiom of Foundation, of which three (and a half) will interest us here. They are:—

The Axiom of Foundation, version 1

The version given in the axiom can be rewritten

$$(\forall a \in \mathbf{U})(a \subseteq w \;\Rightarrow\; a \in w) \qquad \Rightarrow \qquad w = \mathbf{U}$$

There is another way of looking at this statement, so little different that we will call it ...

The Axiom of Foundation, version 1A ($\in$-induction)

A schema: an axiom for every expression $P(x)$

$$(\forall a \in \mathbf{U})\big(\,(\forall x \in a)P(x) \;\Rightarrow\; P(a)\,\big) \qquad \Rightarrow \qquad (\forall a \in \mathbf{U})P(a)$$

The Axiom of Foundation, version 2

There is no sequence $m_0, m_1, m_2, \ldots$ of sets (indexed by $\mathbb{N}$) such that

(i) m_0 is a singleton subset

(ii) for every $i \in \mathbb{N}$ and $x \in m_i$, $\quad x \cap m_{i+1} \neq \varnothing$.

The Axiom of Foundation, version 3

$$(\forall a)\big(\; a = \varnothing \quad \text{or} \quad (\exists x \in a)(x \cap a = \varnothing)\;\big).$$

(Note: in Version 3 we do *not* start with $(\forall a \in \mathbf{U})$ — this is true for all classes a.)

G.2 Remarks

Version 1 is of course the version given at the start of this chapter. If you compare it with the statements of Strong Induction in E.3(ii), you will see that Versions 1 and 1A say that you can do something very much like strong induction on the class of all sets, using the relation $\in$. Hence the subheading "$\in$-induction" above.

▶ E.3

An immediate corollary of Version 2 is that there is no sequence $a_0, a_1, a_2, \ldots$ of sets (indexed by $\mathbb{N}$) such that $a_1 \ni a_1 \ni a_2 \ni \ldots$ This probably accounts for the name of the axiom. To see that this follows from Version 2, define $m_0, m_1, m_2, \ldots$ by $m_i = \{a_i\}$ for all $i \in \mathbb{N}$.

We now prove that these three forms of the Axiom are in fact equivalent (in the presence of the preceding axioms of **VBG**).

Proof that Version 1 $\Rightarrow$ Version 1A. Let w be the class $w = \{\; x \;:\; P(x) \;\}$. ■

Proof that Version 1A $\Rightarrow$ Version 2. Let us call a sequence $m_0, m_1, m_2, \ldots$ satisfying Conditions (i) and (ii) of Version 2 a "bad" sequence. If $m_0 = \{a\}$, it is a bad sequence "starting with" $\{a\}$.

Our aim is to prove that there are no bad sequences; in other words, for every set a, there is no bad sequence starting with $\{a\}$. If we write $P(a)$ for the expression "there is no bad sequence starting with $\{a\}$", then we wish to prove $(\forall a \in \mathbf{U})P(a)$. By virtue of Version 1A, it is enough to prove that

$$(\forall a \in \mathbf{U})\big(\ (\forall x \in a)P(x)\ \Rightarrow\ P(a)\ \big).$$

So let us suppose that a is any set, and prove that $(\forall x \in a)P(x) \Rightarrow P(a)$. This we do by proving that $\neg P(a) \Rightarrow \neg(\forall x \in a)P(x)$, which is the same as $\neg P(a) \Rightarrow (\exists x \in a)\neg P(x)$. So now we assume $\neg P(a)$, that is, that there is some bad sequence starting with $\{a\}$; our aim is to prove that there is some $x \in a$ and a bad sequence starting with $\{x\}$.

Let $m_0, m_1, m_2, \ldots$ be the assumed bad sequence starting with $\{a\}$. We have

(i) $m_0 = \{a\}$.

(ii) For every $i \in \mathbb{N}$ and $x \in m_i$, $\quad x \cap m_{i+1} \neq \varnothing$.

In particular, since $a \cap m_1 \neq \varnothing$, there is an element $b \in a \cap m_1$. Let us say that a sequence $p_0, p_1, p_2, \ldots$ of sets is "poor" if

(1) $p_i \subseteq m_{i+1}$ for all i,

(2) $b \in p_0$, and

(3) for all $i \in \mathbb{N}$ and $x \in p_i$, $\quad x \cap m_{i+2} \subseteq p_{i+1}$.

First note that at least one poor sequence exists: define $p_i = m_{i+1}$ for all i. Therefore the "intersection" of all poor sequences, that is, the sequence $n_0, n_1, n_2, \ldots$ defined by

$$n_i = \bigcap\{\, p_i : p_i \text{ is the } i\text{th term in some poor sequence.}\,\},$$

is in fact a sequence of sets. This sequence can perhaps be more neatly defined

$$x \in n_i \quad \Leftrightarrow \quad x \in p_i \ \text{ for every poor sequence } p_0, p_1, p_2, \ldots\ .$$

We now check that $n_0, n_1, n_2, \ldots$ is poor.

(1) Since the sequence $p_0, p_1, p_2, \ldots$ defined by $p_i = m_{i+1}$ is poor, $n_i \subseteq p_i = m_{i+1}$ for all i.

(2) b is a member of p_0 for every poor sequence, so $b \in n_0$.

3) Let $i \in \mathbb{N}$ and $x \in n_i$. Then for every poor sequence $p_0, p_1, p_2, \ldots,\ x \in p_i$. But then $x \cap m_{i+2} \subseteq p_{i+1}$. Since this is true for all poor sequences, $x \cap m_{i+2} \subseteq n_{i+1}$.

We now show that $n_0, n_1, n_2, \ldots$ is in fact bad. First we show that $n_0 = \{b\}$. By (1) it is enough to show that, if $y \neq b$, then $y \notin n_0$. This is the same as showing that there is a poor sequence $p_0, p_1, p_2, \ldots$ such that $y \notin p_0$. This is easy: set $p_0 = n_0 \smallsetminus \{y\}$ and $p_i = n_i$ for all

$i \geq 1$. Now $p_0 \subseteq n_0 \subseteq m_{i+1}$, so (1) holds. Also $b \in n_0$ and $b \neq y$ so $b \in p_0$ and (2) holds. Suppose that $x \in p_i$. Then $x \in n_i$, so $x \cap m_{i+2} \subseteq n_{i+1} = p_{i+1}$ and (3) holds.

Now suppose that $i \in \mathbb{N}$ and $x \in n_i$. Then, since $n \subseteq m_{i+1}$, $x \in m_{i+1}$. Then $x \cap m_{i+2} \neq \varnothing$, but $x \cap m_{i+2} \subseteq n_{i+1}$ and so $x \cap n_{i+1} \neq \varnothing$.

We have shown that $n_0, n_1, n_2, \ldots$ is a bad sequence starting with $\{b\}$, and $b \in a$, as required. ■

Proof that Version 2 $\Rightarrow$ Version 3. We will prove that Version 3 false $\Rightarrow$ Version 2 false. Our assumption is then that there is a class a such that $a \neq \varnothing$ and $(\forall x \in a)(x \cap a \neq \varnothing)$. We choose $b \in a$ and use it to construct a bad sequence. Define it inductively:

$$\begin{aligned} m_0 &= \{b\} \\ m_{i+1} &= \bigcup \{\, x \cap a : x \in m_i \,\}. \end{aligned}$$

We must first check that each m_i is in fact a set. That m_0 is a set is obvious. Now let $i \in \mathbb{N}$; for every $x \in m_i$, $x \cap a$ is a subclass of a and so a set, in fact a member of $\mathcal{P}(a)$. Thus $\{\, x \cap a : x \in m_i \,\}$ is a subset of $\mathcal{P}(a)$. It is therefore also a set and so is its union.

Now observe that $m_i \subseteq a$ for all i. ($m_0 = \{b\}$ and $b \in a$, so $\{b\} \subseteq a$. Also, for any $y \in m_{i+1}$, there is some $x \in m_i$ such that $y \in x \cap a$ and then $y \in a$; thus $m_{i+1} \subseteq a$.)

Finally check property (ii) of a bad sequence. Let $i \in \mathbb{N}$ and $x \in m_i$. Then $x \in a$, so by assumption, $x \cap a \neq \varnothing$. Thus there is some $y \in x \cap a$. But then $y \in m_{i+1}$. Also $y \in x$. Then $y \in x \cap m_{i+1}$, so $x \cap m_{i+1}$ is nonempty. ■

Proof that Version3 $\Rightarrow$ version1. We will prove that Version 1 false $\Rightarrow$ Version 3 false. Our assumption then is that Version 1 fails, in other words, that there is a class w such that

$$(\forall a \in \mathbf{U})(a \subseteq w \quad \Rightarrow \quad a \in w) \tag{1}$$

but

$$w \neq \mathbf{U}\ . \tag{2}$$

Let b be the complement of w. Then (1) becomes

$$(\forall x)(x \cap b = \varnothing \quad \Rightarrow \quad x \notin b)$$

in other words

$$(\forall x)(x \in b \quad \Rightarrow \quad x \cap b \neq \varnothing) \tag{1'}$$

and in other words again

$$(\forall x \in b)(x \cap b \neq \varnothing)\ . \tag{1''}$$

Also (2) becomes

$$b \neq \varnothing \tag{2'}$$

The last two expressions constitute the negation of Version 3. ■

G.3 Theorem

There is no set x such that $x \in x$.

There are no sets x and y such that $x \in y$ and $y \in x$.

Proof. This is an example of an inductive-type argument using Version 1. I will describe the layout of the argument carefully. Once we see what has to be proved, it is in fact very easy.

To prove the first statement, let $P(a)$ be the expression "$a \notin a$"; we want to prove $(\forall a \in \mathbf{U})P(a)$. By Version 1 it is sufficient to prove $(\forall a \in \mathbf{U})(\,(\forall x \in a)P(x) \;\Rightarrow\; P(a)\,)$. So, let a be any set and assume that $(\forall x \in a)P(x)$, that is, that no member of a is a member of itself. It is required to prove from this that $a \notin a$. But this is easy: if $a \in a$, then a is a member of itself which is a member of itself, contradicting the "inductive" assumption.

For the second statement, let $P(a)$ be the expression "There is no member x of a such that $a \in x$". The argument goes as before. ■

In the section on Ordinal numbers, we will prove an interesting result which follows from the Axiom of Foundation. For the time being we prove a couple of facts about transitivity:

G.4 Theorem

For every set there is a unique smallest transitive set which contains it (as a subset). In more detail: let A be any set. Then there is a transitive set T such that $A \subseteq T$ and, moreover, if T' is any other transitive set such that $A \subseteq T'$ then $T \subseteq T'$. This set T is uniquely defined by A.

(It is called the *transitive closure* of A.)

Proof. We prove that T exists; uniqueness is obvious.

Define a family $(B_i)_{i\in\mathbb{N}}$ of sets, indexed by the natural numbers, as follows:—

$$\begin{aligned} B_0 &= A \\ \text{for each } i \in \mathbb{N}, \quad B_{i+1} &= B_i \cup \left(\bigcup B_i\right) . \end{aligned}$$

(This means that $x \in B_{i+1}$ if and only if either $x \in B_i$ or there is some y such that $x \in y \in B_i$.) Now define $T = \bigcup_{i\in\mathbb{N}} B_i$. It is easy to check that T is transitive and that $A \subseteq T$. It remains to check the minimality condition. So, given transitive T' such that $A \subseteq T'$, we must prove that $T \subseteq T'$; it is enoug to show that every $B_i \subseteq T'$, and this we do by induction over i. Firstly (zerothly?) $B_0 \subseteq T'$ because $B_0 = A$. Now suppose (i-thly?) that $B_i \subseteq T'$; we show tat $B_{i+1} \subseteq T'$. Let $x \in B_i + 1$. Then either $x \in B_i$, in which case $x \in T'$ trivially, or else there is some y such that $x \in y \in B_i$, in whic case $y \in T'$ and then $x \in T'$ by transitivity. ■

G.5 Remark

When dealing with classes of sets with some kind of structure, it is normally of interest to consider functions between those sets which preserve the structure in some obvious way, for examples, continuous functions between topological spaces, linear functions between vector spaces, homomorphisms between groups, order-preserving functions between ordered sets and so on. Now a plain old set has a built-in structure: since all its members are also

sets, it has a defined relation $\in$ between its members. It seems natural to look at functions which preserve this structure. Such a function $f : A \to B$ would have the property that, for all $x, y \in A,\ x \in y \Rightarrow f(x) \in f(y)$. In this context, the following theorem says that two transitive sets are isomorphic if and only if they are the same.

G.6 Theorem

Let A and B be transitive sets and suppose there exists a bijection $f : A \to B$ such that

$$\text{for all } x, y \in A \quad x \in y \ \Leftrightarrow\ f(x) \in f(y)\ ,$$

then $A = B$.

Proof. as an interesting exercise. ■

G.7 VBG7: The Axiom of Formation

The Axiom of Formation tells us that, if the domain of a function is a set, then so is its range. In a little more detail: suppose that f is a function $A \to B$, where B is any class, even $\mathbf{U}$. Writing $P(x, y)$ for the expression '$f(x) = y$', we have

$$(\forall x)(x \in A\ \Rightarrow (\exists! y)P(x, y)).$$

and then the range $f[A]$ would be described thus:

$$y \in f[A] \quad \Leftrightarrow \quad (\exists x)(x \in A\ \wedge\ P(x, y))$$

The Axiom of Formation simply says that, if A is a set, then so is $f[A]$.

An interesting (and important) application of the Axiom of Foundation occurs when a set of sets, indexed by $\mathbb{N}$, is defined inductively. For example, we would expect that we could define the sets

$$\varnothing \quad , \quad \{\varnothing\} \quad , \quad \{\{\varnothing\}\} \quad , \quad \{\{\{\varnothing\}\}\} \quad , \ldots$$

as a sequence inductively somehow and that then the class of all of them:

$$\{\varnothing\ ,\ \{\varnothing\}\ ,\ \{\{\varnothing\}\}\ ,\ \{\{\{\varnothing\}\}\}\ , \ldots\}$$

would be a set. We want to define these sets as a sequence, $S_0, S_1, S_2, \ldots$ say, indexed by $\mathbb{N}$, that is, a function S with domain $\mathbb{N}$. This is easy: we define the function $S : \mathbb{N} \to \mathbf{U}$ by induction by specifying $S_0 = \varnothing$ and $S_{n^+} = \{S_n\}$. The collection of all these sets is of course the range of the function S, and the Axiom of Formation tells us that this is a set.

6. TRANSFINITE ARITHMETIC

A Ordinal numbers

A.1 Definition

An ordinal number is a set α such that

(i) α is transitive (that is, every member of α is also a subset of α).

(ii) For all $x, y \in \alpha$, one of the following hold: $x \in y$, $x = y$ or $y \in x$.

The class of all ordinal numbers will be denoted **Ord**.

Note that, by F5, (i) above is equivalent to

(i′) $x \in \alpha \;\Rightarrow\; x \subset \alpha$.

A.2 Definition: Ordering an ordinal number

Let α be an ordinal. We define the relation $\leq$ on α by

$$x \leq y \qquad \text{if and only if} \qquad x \in y \quad \text{or} \quad x = y\ .$$

A.3 Lemma

This relation is in fact a well order on α.

Proof. As was observed in the note **5**.E.1, it is enough to prove that this relation is antisymmetric and that every nonempty subset of α has a least element. ▶ **5**.E.1

For antisymmetry, suppose that $x \leq y$ and $y \leq x$ in α. By the definition above, either $x \in y$ or $x = y$ and also either $y \in x$ or $y = x$. Of the various cases which arise, the only one permitted by Theorem **5**.G.3 is $x = y$. ▶ **5**.G.3

Now suppose that E is a nonempty subset of α. By Version 3 of the Axiom of Foundation, there is $z \in E$ such that $z \cap E = \varnothing$. We will show that z is the least member of E, as required. We already know that $z \in E$, so it is enough to show that it is a lower bound. Suppose then that x is any member of E. Then $x \notin z$. But then by A.1(ii) above, either $z \in x$ or $z = x$, that is $z \leq x$, as required. ■

A.4 Remarks

Recall the definition of an initial segment in an ordered set: $I(a) = \{\, x : x < a \,\}$. Ordinal numbers have the interesting property that the initial segment of a member is equal to that

member, that is

$$\text{If } \alpha \text{ is an ordinal and } x \in \alpha, \text{ then } I(x) = x.$$

To see this, simply observe that $z \in x \quad \Leftrightarrow \quad z < x \quad \Leftrightarrow \quad z \in I(x)$.

We will be using the idea of order-isomorphism quite often in connection with ordinal numbers. It will be convenient to have a symbol for this relation. If A and B are ordered sets, we will write $A \simeq B$ to mean that A is order-isomorphic to B.

A.5 Lemma

If two ordinal numbers are order-isomorphic, then they are equal.

Proof. By symmetry, it is enough to prove that, if α and β are ordinal numbers and $\alpha \simeq \beta$, then $\alpha \subseteq \beta$. Suppose not. Let $\varphi : \alpha \to \beta$ be the assumed order-isomorphism. Then there is some $x \in \alpha$ such that $x \neq \varphi(x)$ and thus the subset $E = \{\, x : x \neq \varphi(x) \,\}$ of α is nonempty. Since α is well ordered, the set E has a least element, e say. Then $x = \varphi(x)$ for all $x < e$ in α. This means that $I_\alpha(e) = \{\, \varphi(x) : x \in I_\alpha(e) \,\}$. On the other hand, since φ is an order-isomorphism, $\{\, \varphi(x) : x \in I_\alpha(e) \,\} = I_\beta(\varphi(e))$. Therefore $I_\alpha(e) = I_\beta(\varphi(e))$ and so, by the Remark A3 above, $e = \varphi(e)$. But this contradicts the choice of e. ■

A.6 Lemma

Every initial segment of an ordinal number is an ordinal number.

Therefore every member of an ordinal number is an ordinal number.

Proof. Let I be an initial segment of the ordinal number α. First we check that I is transitive. Let $x \in I$. Then $x \in \alpha$ so $x \subseteq \alpha$. But now, for any $z \in x$, we have $z < x$ and so $z \in I$, proving that $x \subseteq I$ and so that I is transitive. Also, if $x, y \in I$, then $x, y \in \alpha$ and so one of $x \in y$, $x = y$, or $y \in x$ must hold. ■

A.7 Theorem

(i) The class **Ord** is transitive (that is, if $\alpha \in$ **Ord**, then $\alpha \subseteq$ **Ord**).

(ii) If $\alpha, \beta \in$ **Ord** then one of $\alpha \in \beta$, $\alpha = \beta$ or $\beta \in \alpha$ must hold.

(iii) Every nonempty subclass of **Ord** has a least element.

► A.6

Proof. (i) is the second statement of A.6.

► 5.E.7

(ii) Here α and β are well ordered sets and so, by the Trichotomy Theorem (**5**.E.7), one of three things must hold. Firstly, $\alpha \simeq \beta$, in which case they are equal, by Lemma A.5 above. Secondly, there is some $y \in \beta$ such that $\alpha \simeq I(y)$. In this case $\alpha = y \in \beta$. Lastly, there is some $x \in \alpha$ such that $\beta \simeq I(x)$, in which case $\beta = x \in \alpha$.

(iii) Let A be a nonempty subclass of **Ord**. Then there is some $\alpha \in A$. If α is the least member of A, then we are done. Otherwise it is enough to show that the set of all members

of A which are $< \alpha$ has a least element. But this is the set of all members of A which are $\in \alpha$ and so is a nonempty subset of the ordinal α; the result follows. ■

A.8 Remarks

Part (iii) of this theorem tells us that **Ord** is a well ordered class.

This theorem also tells us that **Ord** has all the defining properties of an ordinal number — except for one ...

A.9 Theorem

The class **Ord** of all ordinals is a proper class; that is, it is not a set.

However, every proper initial segment of **Ord** is a set (in fact an ordinal number).

Proof. If **Ord** was a set, Parts (i) and (ii) of the previous theorem would mean that it was an ordinal number itself. Then we would have **Ord** $\in$ **Ord**, which is impossible.

Now let J be any proper initial segment of **Ord**. Since it is proper, **Ord** $\setminus J$ is nonempty. Let α be the least member of **Ord** $\setminus J$. Then $J = I(\alpha) = \alpha$. ■

A.10 Theorem

Every well ordered set is order-isomorphic to exactly one ordinal number.

Proof. Since ordinal numbers are well ordered sets, by the Trichotomy Theorem (**5**.E.7) ordinal numbers are of three kinds, ▶ **5**.E.7

(i) Ones which are order-isomorphic to some initial segment $I(w)$ of W.

(ii) Ones which are order-isomorphic to W itself.

(iii) Ordinals α such that W is order-isomorphic to some $I(\xi)$, where $\xi \in \alpha$.

Looking first at (i), note that the initial segments of W form a set; by A4 then, the ordinals of Type (i) form a set. Therefore there exists an ordinal α which is either of Type (ii) or of Type (iii). If it is of Type (ii) we are done. If it is of Type (iii), note that ξ is then an ordinal and $I(\xi) = \xi$, so W is order-isomorphic to the ordinal ξ.

Uniqueness is easy. If W is order-isomorphic to ordinals α and β, then α and β are order-isomorphic to each other and so, by A.5, they are equal. ■ ▶ A.5

A.11 Definition

If A is a well ordered set, then the unique ordinal number to which it is order-isomorphic is called the *order type* of A.

A.12 Lemma

(i) The set $\mathbb{N}$ of natural numbers is an ordinal number.

(ii) Every natural number is an ordinal number. In other words, $\mathbb{N} \subset$ **Ord**.

Remark In the context of ordinal numbers, the set $\mathbb{N}$ is usually denoted ω.

Proof. (i) To prove that ω is transitive, we assume that $m \in n \in \omega$ and prove that then $m \in \omega$. This we do by induction over n. If $n = 0$, the result is vacuously true. Now assume that the result is true for n and prove it for $n+$. We are assuming that $m \in n \cup \{n\}$; but then either $m \in n$, in which case $m \in \mathbb{N}$ by the inductive hypothesis, or else $m = n \in \mathbb{N}$ in which case it is true trivially.

Nowe must prove that, for any $m, n \in \omega$, one of $m \in n$, $m = n$ or $n \in m$ holds. Let us write $m \sim n$ to mean that one of $m \in n$, $m = n$ or $n \in m$ holds. We now wish to prove that $m \sim n$ for all $m, n \in \omega$. The proof goes in several steps.

(1) $\varnothing \in n^+$ for all $n \in \omega$.

Proof is by induction over n. Firstly $\varnothing \in \{\varnothing\} = \varnothing^+$. Now suppose $\varnothing \in n^+$. Then $\varnothing \in n^{++}$ since $n^+ \subseteq n^{++}$.

(2) $\varnothing \sim n$ for all $n \in \omega$. (Corollary of (1).)

(3) If $m \in n$ then $m^+ = n$ or $m^+ \in n$. Proof is by induction over n. If $n = \varnothing$, the result is vacuously true. Now suppose $m \in n^+$; we want to prove that $m^+ = n^+$ or $m^+ \in n^+$. We have $m \in n \cup \{n\}$, so $m \in n$ or $m = n$. If $m = n$ then $m^+ = n^+$. If $m \in n$ then (by the inductive hypothesis), either $m^+ = n$ or $m^+ \in n$. In either case $m^+ \in n^+$.

(4) Now we prove the main result, that $m \sim n$, by induction over n. The case $n = \varnothing$ is give by Step (2) above. Now suppose that $m \sim n$; we want to prove that $m \sim n^+$ also. Let us check the cases. If $m \in n$ then $m \in n^+$ also, since $n \subseteq n^+$. If $m = n$ then $m^+ = n^+$. If $n \in m$ then by (3), either $n^+ = m$ or $n^+ \in m$ and in either case $m \sim n^+$.

(ii) is a corollary of (i). ■

A.13 Theorem

(i) 0 is an ordinal number.

(ii) If α is an ordinal, then so is α^+.

(iii) If A is a set of ordinal numbers, then its union is an ordinal number.

(iv) $\alpha^+ \neq 0$ for any ordinal number α.

(v) If α and β are ordinal numbers,then $\alpha^+ = \beta^+ \Rightarrow \alpha = \beta$.

(vi) α^+ is the successor of α, in the sense that it is the "next" ordinal: there is no ordinal ξ such that $\alpha < \xi < \alpha^+$.

(vii) A corollary of this is that, if α and β are ordinals and $\alpha < \beta$, then $\alpha^+ \leq \beta$.

(viii) Every set A of ordinals has a supremum, that is, an ordinal σ such that

$$\text{for all } \alpha \in A, \qquad \alpha \leq \sigma.$$
$$\text{and, if } \xi \text{ is an ordinal such that, for all } \alpha \in A,\ \alpha \leq \xi \quad \text{then} \quad \sigma \leq \xi.$$

If A has a maximum element, then σ is this maximum element. Otherwise $\sigma = \bigcup A$.

Proof. **(i)** The definition is satisfied vacuously for $0 = \varnothing$.

(ii) and **(iii)** follow immediately from the definition of an ordinal number.

(iv) $\alpha \in \alpha^+$.

(v) If $\alpha^+ = \beta^+$ then $\alpha \in \beta^+ = \beta \cup \{\beta\}$, so $\alpha \in \beta$ or $\alpha = \beta$, that is $\alpha \leq \beta$. But, in the same way, $\beta \leq \alpha$.

(vi) (and **(vii)**) If such a ξ existed, we would have $\alpha \in \xi \in \alpha \cup \{\alpha\}$, so that either $\alpha \in \xi \in \alpha$ or $\alpha \in \xi = \alpha$, both of which contradict the axiom of foundation.

(viii) If σ is the maximum member of A, then of course it is a supremum. Otherwise, set $\sigma = \bigcup A$. It is an ordinal number by (iii). If $\alpha \in A$ then, since A has no maximum element, there is some $\beta \in A$ such that $\alpha < \beta \in A$, in which case $\alpha \in \beta$ so $\alpha \in \sigma$, which is the same as $\alpha < \sigma$. Now suppose that ξ is an ordinal number such that $\alpha \leq \xi$ for all $\alpha \in A$. Since A has no maximum member, then $\alpha < \xi$, that is $\alpha \in \xi$, for all $\alpha \in A$. But then $\bigcup A \subseteq \xi$, which is the same as $\sigma \leq \xi$. ■

A.14 Examples: Some infinite ordinals

As examples of the last theorem, the following are ordinal numbers.

$$\begin{aligned}\omega^+ &= \{0, 1, 2, \ldots\} \cup \{\omega\} \\ \omega^{++} &= \{0, 1, 2, \ldots\} \cup \{\omega, \omega^+\} \\ \omega^{+++} &= \{0, 1, 2, \ldots\} \cup \{\omega, \omega^+, \omega^{++}\}\end{aligned}$$

and so on.

The union of these is called 2ω:

$$2\omega \;=\; \{0, 1, 2, \ldots\} \cup \{\omega, \omega^+, \omega^{++}, \ldots\}$$

And now of course we have

$$(2\omega)^+ \;=\; \{0, 1, 2, \ldots\} \cup \{\omega, \omega^+, \omega^{++}, \ldots\} \cup \{2\omega\}$$

$$(2\omega)^{++} \;=\; \{0, 1, 2, \ldots\} \cup \{\omega, \omega^+, \omega^{++}, \ldots\} \cup \{2\omega, (2\omega)^+\}$$

and so on. Then of course, the union of all these:

$$(3\omega) \;=\; \{0, 1, 2, \ldots\} \cup \{\omega, \omega^+, \omega^{++}, \ldots\} \cup \{2\omega, (2\omega)^+, (2\omega)^{++}, \ldots\}$$

and so on and on.

A.15 Definition

An ordinal number λ is a *limit ordinal* if it is not the successor of any ordinal (that is, not of the form ξ^+ for any ordinal ξ). Otherwise it is called a *successor ordinal* or a *non-limit ordinal*. Thus to say that α is a successor ordinal means that there is some ordinal ξ such that $\alpha = \xi^+$.

Examples of limit ordinals are $0,\ \omega,\ 2\omega,\ 3\omega$.

Examples of non-limit ordinals are: all the nonzero natural numbers ($n = (n-1)^+$) and the ordinals $\omega^+,\ \omega^{++},\ \omega^{+++}$ and so on.

The following useful characterisations of limit ordinals should be proved as an exercise:

(i) An ordinal number λ is a limit ordinal if and only if

$$\xi < \lambda \quad \Rightarrow \quad \xi^+ < \lambda \qquad \text{for all ordinals } \xi.$$

(ii) An ordinal number λ is a limit ordinal if and only if $\lambda = \sup\{\xi : \xi < \lambda\}$.

(This last probably accounts for its name.)

A.16 Lemma: Notes on suprema

We will be using suprema frequently in this chapter. Here are some helpful simple facts.

(i) Let X be a subset of any ordered set. Then $\mu = \sup X$ if and only if the following two things are true:

(a) $\xi \le \mu$ for all $\xi \in X$.

(b) If $\alpha < \mu$ then there is some $\xi \in X$ such that $\alpha < \xi$.

(ii) Let X be a subset of any ordered set. If X has a maximum value, then that is also its supremum (that is, if $\max X$ exists then $\sup X$ exists also and $\sup X = \max X$).

(iii) If the subset X has a supremum and that supremum is a member of X, then it is also the maximum.

(iv) For ordinals only: if X is a nonempty set of ordinals and $\sup X$ is either zero or a successor ordinal, then $\sup X \in X$ and so is a maximum value of X.

(v) Let f be a function $X \to \mathbf{Ord}$, where X is any set. Then $\mu = \sup\{\, f(x) \,:\, x \in X \,\}$ if and only if the following two things are true:

(a) $f(\xi) \le \mu$ for all $\xi \in X$.

(b) If $\alpha < \mu$ then there is some $\xi \in X$ such that $\alpha < f(\xi)$.

▶ 5.C.2

Proof. **(i)**, **(ii)** and **(iii)** follow immediately from the definitions (see **5**.C.2).

(iv) If $\sup X = 0$ then, since $X \neq \varnothing$, we have $X = \{0\}$ and the result is trivially true.

If $\sup X$ is a successor ordinal, $\sup X = \alpha^+$ say, then $\alpha < \alpha^+$ so, by (ib) above, there is some $\xi \in X$ such that $\alpha < \xi$. But then $\alpha^+ = \xi$.

(v) follows immediately from (i). ■

A.17 Theorem: Inductive properties of Ord

(i) (Ordinary induction) Let X be a subclass of **Ord** with the properties:

For any ordinal ξ, $\xi \in X \Rightarrow \xi^+ \in X$

For any limit ordinal λ, $I(\lambda) \subseteq X \Rightarrow \lambda \in X$.

Then $X =$ **Ord**.

(ii) (Strong induction) Let X be a subclass of **Ord** with the property:

For any ordinal ξ, $I(\xi) \subseteq X \Rightarrow \xi \in X$.

Then $X =$ **Ord**.

Proof. **(ii)** is just the statement that **Ord** is well ordered (see **5**.E.3). ▶ **5**.E.3

(i) If $X \neq$ **Ord** then there is an ordinal which is not a member of X. If it is a non-limit ordinal, call it ξ^+ and it violates the first condition; and if it is a limit ordinal, call it λ and it violates the second. ■

A.18 Corollary: Inductive proofs on Ord

(i) (Ordinary induction) Let $P(\xi)$ be a predicate defined on **Ord** with the properties:

For any ordinal ξ, $P(\xi) \Rightarrow P(\xi^+)$.

For any limit ordinal λ, $(\forall \xi < \lambda)(P(\xi)) \Rightarrow P(\lambda)$.

Then $(\forall \xi \in$ **Ord**$)(P(\xi))$.

(ii) (Strong induction) Let $P(\xi)$ be a predicate defined on **Ord** with the property:

For any ordinal ξ, $(\forall \theta < \xi)(P(\theta)) \Rightarrow P(\xi)$.

Then $(\forall \xi \in$ **Ord**$)(P(\xi))$.

A.19 Note

In (ordinary) inductive proofs it is very often the case that the zero case is treated separately from the nonzero limit ordinals. In that case, the structure of the proof is

Let $P(\xi)$ be a predicate defined on **Ord** with the properties:

$P(0)$.

For any ordinal ξ, $P(\xi) \Rightarrow P(\xi^+)$.

For any nonzero limit ordinal λ, $(\forall \xi < \lambda)(P(\xi)) \Rightarrow P(\lambda)$.

Then $(\forall \xi \in \mathbf{Ord})(P(\xi))$.

The various inductive principles over **Ord** above all have corresponding forms for induction over an ordinal number. They, and their proofs, are almost exactly the same, but it is worth listing them here for reference.

A.20 Theorem: Inductive properties of an ordinal number

(i) (Ordinary induction) Let α be an ordinal and X be a subset of α with the properties:

For any $\xi < \alpha$, $\xi \in X \Rightarrow \xi^+ \in X$

For any limit ordinal $\lambda < \alpha$, $I(\lambda) \subseteq X \Rightarrow \lambda \in X$.

Then $X = \alpha$.

(ii) (Strong induction) Let X be a subclass of α with the property:

For any $\xi < \alpha$, $I(\xi) \subseteq X \Rightarrow \xi \in X$.

Then $X = \alpha$.

► 5.E.3

(iii) is just the statement that α is well ordered (see **5**.E.3).

(iv) If $X \neq \alpha$ then there is a member of α, which is therefore an ordinal, which is not a member of X. Consider the first such. If it is a non-limit ordinal, call it ξ^+ and it violates the first condition; and if it is a limit ordinal, call it λ and it violates the second.

A.21 Corollary: Inductive proofs on an ordinal number

(i) (Ordinary induction) Let α be an ordinal and $P(\xi)$ be a predicate defined on α with the properties:

For any ordinal $\xi < \alpha$, $P(\xi) \Rightarrow P(\xi^+)$.

For any limit ordinal $\lambda < \alpha$, $(\forall \xi < \lambda)(P(\xi)) \Rightarrow P(\lambda)$.

Then $(\forall \xi < \alpha)(P(\xi))$.

(ii) (Strong induction) Let α be an ordinal and $P(\xi)$ be a predicate defined on α with the property:

For any ordinal $\xi < \alpha$, $(\forall \theta < \xi)(P(\theta)) \Rightarrow P(\xi)$.

Then $(\forall \xi < \alpha)(P(\xi))$.

A.22 Remark: Comparison of Ord and $\mathbb{N}$

Some of the results of the last few theorems and corollaries look very like the definition of the Natural Numbers by Peano's Axioms. A lot of this is no coincidence.

The natural numbers have the properties

$$0 \in \mathbb{N}$$
$$\xi \in \mathbb{N} \;\Rightarrow\; \xi^+ \in \mathbb{N}$$

and we in fact constructed them as the "smallest" set with these properties — the intersection of all such sets. Similarly, the Ordinal Numbers have the properties

$$0 \in \mathbf{Ord}$$
$$\xi \in \mathbf{Ord} \;\Rightarrow\; \xi^+ \in \mathbf{Ord}$$
$$A \subseteq \mathbf{Ord} \;\Rightarrow \bigcup A \in \mathbf{Ord}$$

and in fact is the smallest class with these properties, that is, the intersection of all such classes (the proof is an easy exercise). Another way of looking at this is that we could have made these properties the original definition of **Ord** and developed everything else from that.

A.23 Definition by induction

In the light of the last remarks we would expect that definition by induction over an ordinal number, or even the class of all ordinals, is allowed; even more so if we compare the remark A.8 with the remarks preceding Theorem **5**.E.9.

▶ A.8
▶ **5**.E.9

Of course **Ord** is a well ordered class, so the kind of inductive definition described in Theorem **5**.E.9 is automatically valid.

▶ **5**.E.9

We also have forms of inductive definition which look more like the strong and weak kinds of inductive definition over $\mathbb{N}$.

The *strong* form is straightforward: for each ordinal α, define $f(\alpha)$ in terms of α and $f|_{I(\alpha)}$. (The point of this is that $f|_{I(\alpha)}$ is that part of the function which has been defined so far.)

Ordinary or *weak* induction is very like ordinary induction over the natural numbers, with just a slight addition to make it "get past" limit ordinals. To define a function f using this kind of induction, separate definitions are given of

$$f(0)$$
$$f(\xi^+) \quad \text{in terms of} \quad \xi \text{ and } f(\xi)$$
$$\text{and, for any limit ordinal } \lambda, \qquad f(\lambda) \quad \text{in terms of } \lambda \text{ and } f|_{I(\lambda)}.$$

These are the simple cases, where there are no parameters. If there are parameters θ from a set T, the weak version looks like this: separate definitions are given of

$$f(\theta, 0)$$
$$f(\theta, \xi^+) \quad \text{in terms of} \quad \theta, \xi \text{ and } f(\theta, \xi)$$
$$\text{and, for any limit ordinal } \lambda, \qquad f(\theta, \lambda) \quad \text{in terms of} \quad \theta, \lambda \text{ and } f|_{T \times I(\lambda)}\ .$$

And the strong version looks like this:

$$\text{For each ordinal } \alpha, \quad \text{define } f(\theta, \alpha) \text{ in terms of } \theta, \alpha \text{ and } f|_{T \times I(\alpha)}.$$

In the majority of cases (I would say), the definition in the limit ordinal case is

$$f(\theta,\lambda) = \bigcup_{\xi<\lambda} f(\theta,\xi) \qquad \text{which is the same as} \qquad f(\theta,\lambda) = \sup_{\xi<\lambda} f(\theta,\xi)$$

or something very like it.

The definitions of addition, multiplication and so on given next are simple examples of the application of this kind of inductive definition. It will be seen that these definitions are really straightforward extensions of the definitions for natural numbers already considered, extended to the ordinals.

We have just considered three kinds of inductive definition over **Ord**. We already know that the first kind is valid. It is a straightforward exercise to use this to prove that the other two kinds are valid also.

We conclude this section with an interesting example of a definition by induction over ordinals.

A.24 Definition: the Cumulative Hierarchy

Define a set $\mathbf{V}_\alpha$, for each ordinal α by induction:—

(i) $\mathbf{V}_0 = \varnothing$.

(ii) For every ordinal α, $\mathbf{V}_{\alpha+1} = \mathcal{P}(\mathbf{V}_\alpha)$ (the power set of $\mathbf{V}_\alpha$) .

(iii) For every nonzero limit ordinal λ, $\mathbf{V}_\lambda = \bigcup\{\, \mathbf{V}_\xi : \xi < \lambda \,\}$.

A.25 Theorem

(i) $\mathbf{V}_\alpha$ is transitive, for every ordinal α.

(ii) If $\alpha < \beta$, then $\mathbf{V}_\alpha \subset \mathbf{V}_\beta$.

(iii) For every set X, there is an ordinal α such that $X \subseteq V_\alpha$ (and then $X \in \mathbf{V}_{\alpha+1}$).

And note that (iii) here says that

$$\mathbf{U} = \bigcup_{\alpha\in\mathbf{Ord}} \mathbf{V}_\alpha \ .$$

Proof. (i) Proof is by induction over α. First notice that $\mathbf{V}_0 = \varnothing$ is transitive (vacuously).

Now suppose that $\mathbf{V}_\alpha$ is transitive and that $x \in y \in \mathbf{V}_{\alpha+1}$. Then $x \in y \subseteq \mathbf{V}_\alpha$ so $x \in \mathbf{V}_\alpha$. But $\mathbf{V}_\alpha$ is transitive (as just assumed) and so $x \subseteq \mathbf{V}_\alpha$. Then $x \in \mathbf{V}_{\alpha+1}$.

Finally suppose that λ is a nonzero limit ordinal, that all the $\mathbf{V}_\xi$ for $\xi < \lambda$ are transitive and that $x \in y \in \mathbf{V}_\lambda$. Then there is some $\xi < \lambda$ such that $y \in \mathbf{V}_\xi$. But then $x \in y \in \mathbf{V}_\xi$ and $\mathbf{V}_\xi$ is transitive (as just assumed); so $x \in \mathbf{V}_\xi$ and then $x \in \mathbf{V}_\lambda$.

(ii) We show first, by induction over β, that if $\alpha \leq \beta$ then $\mathbf{V}_\alpha \leq \mathbf{V}_\beta$. If $\alpha = \beta$ the result is trivial. Now suppose that $\alpha \leq \beta$ and $\mathbf{V}_\alpha \subseteq \mathbf{V}_\beta$; we prove that $\mathbf{V}_\alpha \subseteq \mathbf{V}_{\beta+1}$. Let $x \in \mathbf{V}_\alpha$. Then $x \in \mathbf{V}_\beta \in \mathbf{V}_{\beta+1}$, which is transitive, so $x \in \mathbf{V}_{\beta+1}$. Finally, suppose that $\alpha < \lambda$ and λ is a nonzero limit ordinal. Then $\alpha + 1 < \lambda$ also and $\mathbf{V}_{\lambda+1} \subseteq \mathbf{V}_\lambda$ by the definition of $\mathbf{V}_\lambda$. But $\mathbf{V}_\alpha \subseteq \mathbf{V}_{\alpha+1}$ so $\mathbf{V}_\alpha \subseteq \mathbf{V}_{\alpha+1} \subseteq \mathbf{V}_\lambda$.

To see that, if $\alpha < \beta$ then $\mathbf{V}_\alpha \subset \mathbf{V}_\beta$, it is now enough to show that $\mathbf{V}_\alpha \subset \mathbf{V}_{\alpha+1}$ for all α. But this is easy: we know that $\mathbf{V}_\alpha \subseteq \mathbf{V}_{\alpha+1}$ and the sets cannot be equal because $\mathbf{V}_\alpha \in \mathbf{V}_{\alpha+1}$ but $\mathbf{V}_\alpha \notin \mathbf{V}_\alpha$.

(iii) Let us say "X is in the hierarchy" to mean "there exists some ordinal α such that $X \in \mathbf{V}_\alpha$". We want to prove that every set is in the hierarchy. Using the Axiom of Foundation, Version 1A, it is enough to prove that, for all sets X,

$$\text{all of the members of } X \text{ are in the hierarchy} \quad \Rightarrow \quad X \text{ is in the hierarchy.}$$

Suppose then that X is such a set, that is, for all $x \in X$ there is an ordinal ξ such that $x \in \mathbf{V}_\xi$. Then, for all such x, write $\xi(x)$ for the least ordinal such that $x \in \mathbf{V}_{\xi(x)}$. Now, by the Axiom of Formation, the class $\{\, \xi(x) : x \in X \,\}$ is in fact a set and so, by Theorem A.13(viii), there is an ordinal α such that $\xi(x) \leq \alpha$ for all $x \in X$. Then, for all $x \in X$, $x \in \mathbf{V}_{\xi(x)} \subseteq \mathbf{V}_\alpha$; thus $X \in \mathbf{V}_\alpha$ and so $X \subseteq \mathbf{V}_\alpha$. ■

► A.13

B Ordinal functions and continuity

We consider functions $f : \mathbf{Ord} \to \mathbf{Ord}$, in particular several properties of such functions related to continuity. These are

(1) The function f is *continuous* if, for every nonzero limit ordinal λ and ordinal $\theta < f(\lambda)$ there is an ordinal $\alpha < \lambda$ such that f maps $(\alpha, \lambda]$ into $(\theta, f(\lambda)]$.

(2) The function f has the *sup property* if, for every nonzero limit ordinal λ,

$$\sup_{\xi<\lambda} f(\xi) \;=\; f(\lambda)\,.$$

(3) The function f has the *minsup property* if, for every nonzero limit ordinal λ,

$$\min_{\alpha<\lambda} \sup_{\alpha<\xi<\lambda} f(\xi) \;=\; f(\lambda)\,.$$

(4) The function f has the *intermediate value* or *IV property* if, for all ordinals α_1, α_2 and γ such that $\alpha_1 < \alpha_2$ and

$$f(\alpha_1) \;\le\; \gamma \;<\; f(\alpha_2)\,.$$

Then there is an ordinal β such that

$$\alpha_1 \le \beta \;<\; \alpha_2 \tag{–1}$$

$$\text{and} \qquad f(\beta) \;\le\; \gamma \;<\; f(\beta^+)\,. \tag{–2}$$

We show here that

$$\textit{continuity} \;\Rightarrow\; \text{the } \textit{minsup property} \;\Rightarrow\; \text{the } \textit{IV property}\,.$$

In general, no other implications exist between these properties.

However, in the case of weakly order-preserving functions, all four properties are equivalent.

B.1 Lemma

Suppose that f has the minsup property. Then, for any nonzero limit ordinal λ, there is an ordinal $\alpha_0 < \lambda$ such that

$$\sup_{\alpha<\xi<\lambda} f(\xi) \;=\; f(\lambda) \qquad \text{for all } \alpha \ge \alpha_0 \tag{–1}$$

$$\text{and} \quad \sup_{\alpha<\xi<\lambda} f(\xi) \;>\; f(\lambda) \qquad \text{for all } \alpha < \alpha_0\,. \tag{–2}$$

Proof of (–1). From the minsup property, there is some ordinal $\alpha_0 < \lambda$ such that

$$\sup_{\alpha<\xi<\lambda} f(\xi) \;=\; f(\lambda)\,; \tag{–3}$$

suppose that α_0 is the least such. Then, for $\alpha_0 \le \alpha < \lambda$, by (–3), we have $(\alpha, \lambda) \subseteq (\alpha_0, \lambda)$ so

$$\sup_{\alpha<\xi<\lambda} f(\xi) \;\le\; \sup_{\alpha_0<\xi<\lambda} f(\xi) \;=\; f(\lambda)\,.$$

On the other hand, $\sup_{\alpha<\xi<\lambda} f(\xi) \ge f(\lambda)$ by the choice of α.

Proof of (–2). Suppose that $\alpha < \alpha_0$. Then $(\alpha, \lambda) \supseteq (\alpha_0, \lambda)$ so, by (–3), $\sup_{\alpha<\xi<\lambda} f(\xi) \ge f(\lambda)$. On the other hand, $\sup_{\alpha<\xi<\lambda} f(\xi) \ne f(\lambda)$ by the choice of α_0. ■

B.2 Lemma

(i) *Continuity* $\Rightarrow$ the *minsup property.*

(ii) The *minsup property* $\Rightarrow$ the *IV property.*

Proof. (i) Let f be a continuous function.

Let λ be a limit ordinal and θ any ordinal such that $\theta < f(\lambda)$. Then there is $\beta < \lambda$ such that f maps $(\beta, \lambda]$ into $(\theta, f(\lambda)]$.

Given any α such that $\beta \le \alpha < \lambda$, we have $(\alpha, \lambda] \subseteq (\beta, \lambda]$, so f maps $(\alpha, \lambda]$ into $(\theta, f(\lambda)]$, that is,

$$\alpha < \xi \le \lambda \;\Rightarrow\; \theta < f(\xi) \le f(\lambda)\,.$$

On the other hand, given any $\nu < f(\lambda)$, there is $\mu < \lambda$ such that f maps $(\mu, \lambda]$ into $(\nu, f(\lambda)]$. But, since λ is a limit ordinal, $\mu < \lambda$ and $\alpha < \lambda$, we have $\mu^+ < \lambda$ and $\alpha^+ < \lambda$, so, writing $\xi = \max\{\,\mu^+\,,\,\alpha^+\,\}$, we have $\xi \in (\alpha, \lambda)$ and $\xi \in (\mu, \lambda]$ so $f(\xi) > \nu$. This shows that

$$f(\lambda) \;=\; \sup_{\alpha<\xi<\lambda} f(\xi) \qquad \text{for any } \alpha \ge \beta\,.$$

But if $\alpha < \beta$ then the index set is enlarged so (trivially)

$$f(\lambda) \;\le\; \sup_{\alpha<\xi<\lambda} f(\xi)\,.$$

This shows that

$$f(\lambda) \;=\; \min_{\alpha<\lambda} \sup_{\alpha<\xi<\lambda} f(\xi)\,. \qquad ■$$

(ii) Suppose that f has the minsup property and that α_1, α_2 and θ are ordinals such that $\alpha_1 < \alpha_2$ and

$$f(\alpha_1) \;\le\; \gamma \;<\; f(\alpha_2)\,;$$

we want to show that there is an ordinal β such that (–1) and (–2) above hold.

Let μ be the least ordinal such that $\mu > \alpha_1$ and $f(\mu) > \gamma$. There is at least one such ordinal, namely α_2, so μ exists and $\mu \le \alpha_2$.

Suppose that μ is a limit ordinal. Since $\mu > \alpha_1$ it is a nonzero limit ordinal, so, by the previous lemma, there is an ordinal $\alpha_0 < \mu$ such that

$$\sup_{\alpha<\xi<\mu} f(\xi) \;=\; f(\mu) \qquad \text{for all } \alpha \ge \alpha_0\,.$$

Setting $\alpha' = \max\{\alpha_0, \gamma\}$, we have $\alpha' \geq \alpha_0$,

$$\sup_{\alpha' < \xi < \mu} f(\xi) \;=\; f(\mu) \qquad \text{for all } \alpha \geq \alpha_0 \,.$$

and $\gamma < \mu$, so there is an ordinal ξ such that $\alpha' < \xi < \mu$ and $f(\xi) > \gamma$, contradicting the choice of μ.

So μ is not a limit ordinal: there is β such that $\mu = \beta^+$. Then $\beta \geq \alpha_1$ since $\mu > \alpha_1$, $\beta < \mu \leq \alpha_2$, $f(\beta) \leq \gamma$ by choice of μ and $f(\beta_+) = f(\mu) > \gamma$.

B.3 Lemma

(i) The *IV property* $\not\Rightarrow$ the *minsup property*.

(ii) The *IV property* $\not\Rightarrow$ *continuity*.

(iii) *Continuity* $\not\Rightarrow$ the *sup property*.

(iv) The *minsup property* $\not\Rightarrow$ the *sup property*.

(v) The *IV property* $\not\Rightarrow$ the *sup property*.

(vi) The *sup property* $\not\Rightarrow$ the *IV property*.

(vii) The *sup property* $\not\Rightarrow$ *continuity*.

(viii) The *sup property* $\not\Rightarrow$ the *minsup property*.

Finding counterexample functions to show these is left as an interesting exercise.

B.4 Lemma

Let f be a weakly order-preserving function. Then all four properties above are equivalent for f.

Proof. We already know from the previous section that

$$\textit{Continuity} \;\Rightarrow\; \text{the } \textit{minsup property} \;\Rightarrow\; \text{the } \textit{IV property}$$

so now it is enough to prove that

$$\text{the } \textit{IV property} \;\Rightarrow\; \text{the } \textit{sup property} \;\Rightarrow\; \textit{continuity}$$

under the added condition that f is weakly increasing.

(1) the *IV property* + *weakly increasing* $\Rightarrow$ the *sup property*.

Suppose that f has the IV property and is weakly increasing. Let λ be a nonzero limit ordinal. Then, for all $\xi < \lambda$, $f(\xi) \leq f(\lambda)$ so $\sup_{\xi<\lambda} f(\xi) \leq f(\lambda)$.

Now suppose that $\theta < f(\lambda)$: we want to show that $\sup_{\xi<\lambda} f(\xi) \geq \theta$. If $f(0) > \theta$ then of course $\sup_{\xi<\lambda} f(\xi) \geq f(\theta) > \theta$ and we are done. Otherwise $f(0) \leq \theta < f(\lambda)$ and there is β

such that $\beta < \lambda$ and $f(\beta) \le \theta < f(\beta^+)$. But, since λ is a limit ordianal, $\beta^+ < \lambda$, so (from $f(\beta^+) > \theta$) we have $\sup_{\xi<\lambda} f(\xi) > \theta$.

(2) The *sup property + weakly increasing* $\Rightarrow$ *continuity*.

Suppose that f has the sup property and is weakly increasing. Let λ be a nonzero ordinal and $\theta < \lambda$. We want to show that there is $\alpha < \lambda$ such that f maps $\alpha, \lambda]$ into $(\theta, f(\lambda)]$.

Since $\sup_{\xi<\lambda} f(\xi) = f(\lambda)$ and $\theta < f(\lambda)$, there is $\alpha < \lambda$ such that $\theta < f(\alpha) \le f(\lambda)$. But then

$$\xi \in (\alpha, \lambda] \;\Rightarrow\; \alpha < \xi \le \lambda \;\Rightarrow\; f(\alpha) \le f(\xi) \le f(\lambda) \;\Rightarrow\; f(\xi) \in (\theta, f(\lambda)]\,,$$

since $\theta < f(\alpha)$. ■

B.5 A useful lemma

Suppose that f is a function **Ord** $\to$ **Ord** which is continuous *or* has the minsup property *or* has the sup property. Then

(i) if also

$$f(\xi) \;\le f(\xi^+) \qquad \text{for all ordinals } \xi$$

then f is weakly increasing,

(ii) if also

$$f(\xi) \;< f(\xi^+) \qquad \text{for all ordinals } \xi$$

then f is strictly increasing.

Proof. The proof of **(i)** is given; the proof of **(ii)** is the same.

First let us note that, if f has the minsup property, then by Lemma B.1, for any nonzero limit ordinal λ, there exists an ordinal $\alpha < \lambda$ such that ▶ B.1

$$\sup_{\alpha<\xi<\lambda} f(\xi) \;=\; f(\lambda)\,. \tag{–1}$$

But the same is obviously true if f has the sup property (put $\alpha = 0$) and continuity implies the minsup condition, so (–1) holds in any case.

Suppose then that f is not weakly increasing. Then there are ordinals $\alpha < \beta$ such that $f(\alpha) > f(\beta)$. Let β be the least ordinal for which there exists $\alpha < \beta$ such that $f(\alpha) > f(\beta)$. Then β must be a limit ordinal, for otherwise there would be ξ such that $\beta = \xi^+$ and then $f(\xi) \le f(\beta) < f(\alpha)$ with $\xi \ge \alpha$ and then $\xi > \alpha$, contradicting the choice of β.

So β is a nonzero limit ordinal (since $\beta > \alpha$) and so, as noted above, there is $\theta < \beta$ such that

$$\sup_{\theta<\xi<\beta} f(\xi) = f(\beta)\,. \tag{–2}$$

We may assume without loss of generality that $\theta \ge \alpha$. Then $\alpha < \theta^+ < \beta$ and $f(\theta^+) \ge f(\alpha) > f(\beta)$ contradicting (–2). ■

B.6 Note

The lemma above does not hold if the IV property is used instead of the others. To see this, consider the function f defined by

$$f(\xi) \;=\; \begin{cases} \xi & \text{if } \xi < \omega, \\ 0 & \text{otherwise}. \end{cases}$$

B.7 Lemma

If a function f : **Ord** $\to$ **Ord** is weakly order-preserving and continuous, then, for any nonempty set X of ordinals

$$\sup\{\, f(\xi) \,:\, \xi \in X \,\} \;=\; f(\sup X)\,. \tag{–1}$$

Proof. Set $\mu = \sup X$.

If μ is 0 or a successor ordinal, then it is the maximum value of X and then

$$\max\{\, f(\xi) \,:\, \xi \in X \,\} \;=\; f(\max X)$$

so (–1) follows.

Now suppose that μ is a nonzero limit ordinal.

For any $\xi \in X$, $\xi \le \mu$ so, since f is order-preserving, $f(\xi) \le f(\mu)$; it follows that

$$\sup\{\, f(\xi) \,:\, \xi \in X \,\} \;\le\; f(\mu) \;=\; f(\sup X)\,.$$

To see that equality holds, we suppose that $\sup\{\, f(\xi) \,:\, \xi \in X \,\} < f(\mu)$ and deduce a contradiction. Put

$$\theta = \sup\{\, f(\xi) \,:\, \xi \in X \,\} \qquad \text{so that} \qquad \theta < f(\sup X)\,. \tag{–2}$$

By continuity, there is an $\alpha < \sup X$ such that f maps $(\alpha, \sup X]$ into $(\theta, f(\sup X)]$, that is

$$\theta \;<\; f(\xi) \;\le\; f(\sup X) \qquad \text{for all } \xi \text{ such that } \alpha < \xi \le \sup X\,. \tag{–3}$$

But, since $\alpha < \sup X$, there is some $\xi \in X$ such that $\alpha < \xi$. Then

$$\begin{aligned} \theta \;&\ge\; f(\xi) && \text{by (–2)} \\ &>\; \theta && \text{by (–3)}, \end{aligned}$$

the contradiction. ■

C Arithmetic of ordinal numbers

C.1 Definition: Addition of ordinals

Let α and β be two ordinals. We define the ordinal $\alpha + \beta$ by induction over β:

$$\alpha + 0 = \alpha.$$
$$\alpha + \beta^+ = (\alpha + \beta)^+$$

and, for any nonzero limit ordinal λ,

$$\alpha + \lambda = \sup\{\alpha + \xi : \xi < \lambda\}$$

It is clear from the first two lines of this definition that the new definition of addition coincides with the old one on the natural numbers; in other words, our new definition extends the old one.

We observe that

$$\omega + 1 = \omega + 0^+ = (\omega + 0)^+ = \omega^+,$$
$$\omega + 2 = \omega + 1^+ = (\omega + 1)^+ = \omega^{++} \quad \text{and so on.}$$

also

$$\begin{aligned} \omega + \omega &= \sup\{\omega + \xi : \xi < \omega\} \\ &= \sup\{\omega + 0,\ \omega + 1,\ \omega + 2,\ \ldots\} \\ &= \bigcup\{\omega,\ \omega^+,\ \omega^{++},\ \ldots\} \\ &= 2\omega \end{aligned}$$

$$3\omega = 2\omega + \omega$$

and so on.

By an easy induction,

$$0 + \alpha \ = \ \alpha \qquad \text{for every ordinal } \alpha\,.$$

All this is as expected so far, but note well that addition is not commutative. For example $1 + \omega = \omega$. To see this,

$$1 + \omega \ = \ \sup\{1 + \xi : \xi < \omega\} \ = \sup\{1,\ 2,\ 3,\ \ldots\} = \omega \ \neq \ \omega + 1\,.$$

This also example shows that neither is it cancellative ($1 + \omega = 0 + \omega$). There are a number of other standard algebraic properties that we might expect of addition that fail for ordinal addition, so one should tread with care when doing ordinal arithmetic.

C.2 Theorem: Order properties of addition

(i) Addition is strict order preserving in its second argument: $\xi < \eta \quad \Rightarrow \quad \alpha + \xi < \alpha + \eta$.

It is weak order preserving in its first argument: $\xi \le \eta \quad \Rightarrow \quad \xi + \alpha \le \eta + \alpha$.

(It is not strict order preserving in its first argument: there are examples of ordinals ξ, η and α such that $\xi < \eta$ but $\xi + \alpha = \eta + \alpha$.)

(ii) Addition is continuous in its second argument, and so ...

(ii′) For any ordinal α and any nonempty set X of ordinals, $\sup\{\, \alpha + \xi \,:\, \xi \in X \,\} = \alpha + \sup X$.

(iii) If α and β are ordinals and $\alpha \le \beta$, then there is an ordinal γ such that $\alpha + \gamma = \beta$.

(This rule does not hold on the other side: there may not be any γ such that $\gamma + \alpha = \beta$; for example, there is no γ such that $\gamma + 1 = \omega$.)

Proof. (i) That addition is strictly order-preserving in its second argument follows from
▶ B.5 Lemma B.5.

To see that it is weakly order-preserving in its first argument, suppose that α, ξ and η are ordinals and that $\xi \le \eta$; we want to show that $\xi + \alpha \le \eta + \alpha$. This we do by induction over α.

If $\alpha = 0$ the result is trivial. If α is a non-limit ordinal, $\alpha = \theta^+$ say, then

$$\xi + \alpha = \xi + \theta^+ = (\xi + \theta)^+ \le (\eta + \theta)^+ = \eta + \theta^+ = \eta + \alpha \ .$$

Now suppose that α is a non zero limit ordinal. Then, for any $\theta < \alpha$ we have

$$\xi + \theta \le \eta + \theta \le \sup\{\eta + \zeta : \zeta < \alpha\} = \eta + \alpha \ .$$

Therefore $\xi + \alpha = \sup\{\xi + \zeta : \zeta < \alpha\} \le \eta + \alpha$.

▶ B.4 **(ii)** This is by its definition and Lemma B.4.

▶ B.7 **(ii′)** By B.7.

(iii) Addition is weakly order-preserving in its first argument, so we have $\beta < \beta^+ = 0 + \beta^+ \le \alpha + \beta^+$, so $\alpha + 0 \le \beta < \alpha + \beta^+$. Applying the IV property to the second argument, there is γ such that $0 \le \gamma < \beta^+$ and $\alpha + \gamma \le \beta < \alpha + \gamma^+$. But $\alpha + \gamma^+ = (\alpha + \gamma)^+$ and so $\beta = \alpha + \gamma$. ■

C.3 Theorem: Algebraic properties of addition

(i) Addition is associative: for any ordinals α, β and γ, $\quad (\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$.

(ii) Zero is a true zero: for any ordinal α, $\quad 0 + \alpha = \alpha + 0 = \alpha$.

(iii) Addition is cancellative on the left: $\quad \alpha + \xi = \alpha + \eta \quad \Rightarrow \quad \xi = \eta$,

(but not on the right: there are examples of ordinals ξ, η and α such that $\xi + \alpha = \zeta + \alpha$ but $\xi \ne \zeta$).

Proof. (i) We prove this by induction over γ. In the case $\gamma = 0$ we have

$$(\alpha + \beta) + 0 = \alpha + \beta = \alpha + (\beta + 0) \ .$$

If γ is a non-limit ordinal, $\gamma = \theta^+$, say, we have

$$(\alpha + \beta) + \theta^+ = ((\alpha + \beta) + \theta)^+ = (\alpha + (\beta + \theta))^+ = \alpha + (\beta + \theta)^+ = \alpha + (\beta + \theta^+) \ .$$

If γ is a nonzero limit ordinal, we have

$$\begin{aligned} (\alpha + \beta) + \gamma &= \sup\{(\alpha + \beta) + \xi : \xi < \gamma\} && \text{by definition of addition} \\ &= \sup\{\alpha + (\beta + \xi) : \xi < \gamma\} && \text{by inductive hypothesis} \\ &= \alpha + \sup\{\beta + \xi : \xi < \gamma\} && \text{by C.2(ii), proved above} \\ &= \alpha + (\beta + \gamma) && \text{by the definition of addition again.} \end{aligned}$$

■

(ii) $\alpha + 0 = \alpha$ by definition. We prove that $0 + \alpha = \alpha$ by induction over α. In the case $\alpha = 0$ we have $0 + 0 = 0$. If α is a non-limit ordinal, $\alpha = \theta^+$, say, we have $0 + \alpha = 0 + \theta^+ = (0 + \theta)^+ = \theta^+ = \alpha$. Finally, if α is a nonzero limit ordinal, $0 + \alpha = \sup\{0 + \xi : \xi < \alpha\} = \sup\{\xi : \xi < \alpha\} = \alpha$.

(iii) We prove this in the form $\xi \neq \eta \quad \Rightarrow \quad \alpha + \xi \neq \alpha + \eta$. If $\xi \neq \eta$ then either $\xi < \eta$ or $\xi > \eta$. But then by (iv), proved above, either $\alpha + \xi < \alpha + \eta$ or $\alpha + xi > \alpha + \eta$.

C.4 Definition: Multiplication of ordinals

We define $\alpha\beta$ by induction over β:

$$\begin{aligned} \alpha 0 &= 0 \ . \\ \alpha\beta^+ &= \alpha\beta + \alpha \end{aligned}$$

and, for any nonzero limit ordinal λ,

$$\beta\lambda = \sup\{\beta\xi : \xi < \lambda\}$$

Once again we observe that this definition extends the usual definition of multiplication from the natural numbers to all the ordinals.

We see that

$$\begin{aligned} \alpha 1 &= \alpha 0^+ = \alpha 0 + \alpha = 0 + \alpha = \alpha, \\ \alpha 2 &= \alpha 1^+ = \alpha 1 + \alpha = \alpha + \alpha, \\ \alpha 3 &= \alpha 2^+ = \alpha 2 + \alpha = \alpha + \alpha + \alpha, \end{aligned}$$

and so on, as you would expect. But once again there are a few surprises, for instance,

$$2\omega = \sup\{2\xi : \xi < \omega\} = \sup\{2,\ 4,\ 6,\ \ldots\} = \omega$$

so $2\omega \neq \omega 2$ and multiplication is not commutative.

C.5 Theorem: Order properties of multiplication

(i) Multiplication is strict order preserving on the right: for any ordinal numbers ξ, η and β,

$$\text{if} \quad \beta \neq 0 \quad \text{and} \quad \xi < \eta \qquad \text{then} \qquad \beta\xi < \beta\eta$$

and weak order preserving on the left: for any ordinal numbers α (zero or not), ξ and η,

$$\text{if} \quad \xi \leq \eta \qquad \text{then} \qquad \xi\alpha \leq \eta\alpha \ .$$

(ii) Multiplication is continuous on the right, and so ...

(ii′) For any ordinal α and any nonempty set X of ordinals, $\sup\{\, \alpha\xi \,:\, \xi \in X \,\} \;=\; \alpha \sup X$.

Proof. (i) That multiplication is strictly order-preserving in its second argument follows from Lemma B.5. ▶ B.5

Now we show that multiplication is weak order-preserving on the left, $\xi \leq \eta \;\Rightarrow\; \xi\beta \leq \eta\beta$, by induction over β.

If $\beta = 0$, both sides are zero and the result is trivial.

If β is a non-limit ordinal, $\beta = \theta^+$, say, we have

$$\xi\beta \;=\; \xi\theta^+ \;=\; \xi\theta + \xi \;\leq\; \eta\theta + \eta \;=\; \eta\theta^+ \;=\; \eta\beta$$

(we are using here the fact that addition is order-preserving on both sides, proved above).

Suppose α is a limit ordinal. For any $\theta < \beta$ we have $\xi\theta \leq \eta\theta \leq \sup\{\eta\theta : \theta < \beta\} = \eta\beta$. As this is true for any $\theta < \beta$, we have $\xi\beta = \sup\{\xi\theta : \theta < \beta\} < \eta\beta$.

(ii) This is by its definition and Lemma B.4. ▶ B.4

(ii′) By B.7. ▶ B.7 ■

C.6 Theorem: Algebraic properties of multiplication

(i) Multiplication is associative: for any ordinal numbers α, β and γ, $(\alpha\beta)\gamma = \alpha(\beta\gamma)$.

(ii) Zero behaves the way you would expect: for any ordinal number α, $\alpha 0 = 0\alpha = 0$.

(iii) One behaves the way you would expect: for any ordinal number α, $\alpha 1 = 1\alpha = \alpha$.

(iv) Multiplication is not commutative: there are ordinals α and β such that $\alpha\beta \neq \beta\alpha$.

(v) Multiplication distributes on the left but not on the right: for any ordinal numbers α, β and γ,

$$\alpha(\beta + \gamma) \;=\; \alpha\beta + \alpha\gamma$$

however there are ordinal numbers α, β and γ such that

$$(\alpha + \beta)\gamma \;\neq\; \alpha\gamma + \beta\gamma \ .$$

(vi) Multiplication is cancellative on the left but not on the right: for any ordinal numbers α, ξ and η,

$$\alpha \neq 0 \quad \text{and} \quad \alpha\xi = \alpha\eta \qquad \Rightarrow \qquad \xi = \eta \ .$$

however there are ordinal numbers α, ξ and η such that

$$\alpha \neq 0 \quad \text{and} \quad \xi\alpha = \eta\alpha \qquad \text{but} \qquad \xi \neq \eta$$

(vii) The division algorithm for ordinals. Let α and β be ordinals with $\alpha \neq 0$. Then there are unique ordinals δ and ρ such that

$$\beta = \alpha\delta + \rho \qquad \text{and} \qquad \rho < \alpha.$$

Proof. (ii) We need only prove that $0\beta = 0$, since $\beta 0 = 0$ by definition. Proof is by induction over β and is very easy as you would expect. In the case $\beta = 0$ we have $0\beta = 0.0 = 0$. Now suppose that $\beta = \theta^+$. Then $0\beta = 0\beta^+ = 0\beta + 0 = 0 + 0 = 0$. And if β is a nonzero limit ordinal, $0\beta = \sup\{0\xi : \xi < \beta\} = \sup\{0\} = 0$.

(iii) To see that $\alpha 1 = \alpha$: $\alpha = \alpha 0^+ = \alpha 0 + \alpha = 0 + \alpha = \alpha$.

We prove that $1\beta = \beta$ as usual by induction over β. In the case $\beta = 0$ this is just $1.0 = 0$, which we know already. If $\beta = \theta^+$ then $1\beta = 1\theta^+ = 1\theta + 1 = \theta + 1 = \theta^+ = \beta$. And if β is a limit ordinal, then $1\beta = \sup\{1\xi : \xi < \beta\} = \sup\{\xi : \xi < \beta\} = \beta$.

(iv) We have seen above that $2\omega = \omega \neq \omega 2$.

(v) We prove that $\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$ by induction over γ. This follows the by now well-trodden path.

In the case $\gamma = 0$ we have $\alpha(\beta + 0) = \alpha\beta = \alpha\beta + 0 = \alpha\beta + \alpha 0$.

If $\gamma = \theta^+$, we have

$$\alpha(\beta + \gamma^+) \ = \ \alpha(\beta + \gamma)^+ \ = \ \alpha(\beta + \gamma) + \alpha \ = \ \alpha\beta + \alpha\gamma + \alpha \ = \ \alpha\beta + \alpha\gamma^+ \, .$$

And if β is a nonzero limit ordinal,

$$\begin{aligned}
\alpha(\beta + \gamma) &= \alpha \sup\{\beta + \xi : \xi < \gamma\} && \text{by definition of addition} \\
&= \sup\{\alpha(\beta + \xi) : \xi < \gamma\} && \text{by continuity of multiplication} \\
&= \sup\{\alpha\beta + \alpha\xi : \xi < \gamma\} && \text{by inductive hypothesis} \\
&= \alpha\beta + \sup\{\alpha\xi : \xi < \gamma\} && \text{by continuity of addition} \\
&= \alpha\gamma + \alpha\gamma && \text{by definition of multiplication}
\end{aligned}$$

To see that multiplication does not distribute on the other side, observe that $(1 + 1)\omega = 2\omega = \omega$ but $1\omega + 1\omega = \omega + \omega = \omega 2$ and we already know that these are not equal.

(vi) We prove this result in the form: $\alpha \neq 0 \ \wedge \ \xi \neq \eta \quad \Rightarrow \quad \alpha\xi \neq \alpha\eta$.

Then either $\xi < \eta$ or $\xi > \eta$, in which case $\alpha\xi < \alpha\eta$ or $\alpha\xi > \alpha\eta$, since multiplication is strict order preserving on that side.

To see that multiplication is not cancellative on the other side, observe that $2\omega = 1\omega$.

(i) We prove that $(\alpha\beta)\gamma = \alpha(\beta\gamma)$ by induction over γ.

In the case $\gamma = 0$ we have $(\alpha\beta)0 = 0 = \alpha 0 = \alpha(\beta 0)$.

Now suppose that γ is a non-limit ordinal, $\gamma = \theta^+$, say. Then

$$\begin{aligned}
(\alpha\beta)\gamma &= (\alpha\beta)\gamma^+ && \\
&= (\alpha\beta)\theta + \alpha\beta && \text{by definition of multiplication} \\
&= \alpha(\beta\theta) + \alpha\beta && \text{by inductive hypothesis} \\
&= \alpha(\beta\theta + \beta) && \text{by distribution, proved above} \\
&= \alpha(\beta\theta^+) && \text{by definition of multiplication} \\
&= \alpha(\beta\gamma) &&
\end{aligned}$$

Finally suppose that α is a limit ordinal. Then, using continuity and the inductive hypothesis,

$$(\alpha\beta)\gamma \;=\; \sup\{\,(\alpha\beta)\xi \,:\, \xi < \gamma\,\} \;=\; \sup\{\,\alpha(\beta\xi) : \xi < \gamma\,\} \;=\; \alpha\sup\{\,\beta\xi \,:\, \xi < \gamma\,\} \;=\; \alpha(\beta\gamma).$$

(vii) Given $\alpha > 1$ and β, we have $0\alpha \leq \beta$ and $\beta^+\alpha > \beta$. Also multiplication is continuous on the right, so by the Intermediate Value property (B.4), there is an ordinal δ such that $\delta\alpha \leq \beta < \delta^+\alpha$.

▶ B.4

Now, writing $\beta = \delta\alpha + \rho$, we have

$$\delta\alpha \;\leq\; \delta\alpha + \rho \;<\; \delta\alpha + \alpha$$

and so, by strict order of addition on the right, $0 \leq \rho < \alpha$, as required. ■

C.7 Definition: Exponentiation of ordinal numbers

Let α and β be ordinal numbers. We define α^β by induction over β:

$$\begin{aligned}
\alpha^0 &= 1\ , \\
\alpha^{\beta^+} &= \alpha^\beta\alpha
\end{aligned}$$

and, for any limit ordinal λ,

$$\alpha^\lambda = \sup\{\alpha^\xi : \xi < \lambda\}\ .$$

Once again we observe that this definition extends the usual definition of exponentiation from the natural numbers to all the ordinals.

We see that

$$\begin{aligned}\alpha^1 &= \alpha^{0^+} = \alpha^0\alpha = 1\alpha = \alpha,\\ \alpha^2 &= \alpha^{1^+} = \alpha^1.\alpha = \alpha\alpha,\\ \alpha^3 &= \alpha^{2^+} = \alpha^2\alpha = \alpha\alpha\alpha,\end{aligned}$$

and so on, as you would expect. But once again there are a few surprises, for instance,

$$2^\omega = \sup\{2^\xi : \xi < \omega\} = \sup\{1,\ 2,\ 4,\ 8,\ \ldots\} = \omega$$

.

C.8 Theorem: Order properties of exponentiation

(i) A couple of special cases: when $\alpha = 0$ we have

$$0^\beta = \begin{cases} 1 & \text{if } \beta = 0\ , \\ 0 & \text{otherwise.}\end{cases}$$

and when $\alpha = 1$ we have $1^\beta = 1$ for all β.

(ii) Exponentiation is (mostly) strict order-preserving in its second argument:

$$\text{If } \alpha \geq 2 \text{ and } \xi < \eta, \text{ then } \alpha^\xi < \alpha^\eta.$$

(iii) Exponentiation is (mostly) weakly order-preserving in both arguments.

$$\text{If } \beta \leq \gamma \text{ then } \alpha^\beta \leq \alpha^\gamma \qquad \text{(except in the case that } \alpha = 0,\ \beta = 0 \text{ and } \gamma \neq 0.)$$

and

$$\text{if } \alpha \leq \beta \text{ then } \alpha^\gamma \leq \beta^\gamma \qquad \text{(always).}$$

(iv) Exponentiation is continuous in its second argument, so ...

(iv′) For any ordinal α and nonempty set X of ordinals, and provided that it is not the case that $\alpha = 0$ and X contains both 0 and some nonzero member,

$$\alpha^{\sup X} = \sup\{\alpha^\xi : \xi \in X\}.$$

Proof. (i) This follows directly from the definition.

(ii) That follows from Lemma B.5. ▶ B.5

(iii) That it is weakly order-preserving in its second argument follows from (i) and (ii) above.

To see that it is weakly order-preserving in its first argument, we assume that $\alpha \leq \beta$ and show that $\alpha^\gamma \leq \beta^\gamma$, as usual by induction over γ. Firstly $\alpha^0 = 1 = \beta^0$. Next $\alpha^{\theta^+} = \alpha^\theta\alpha \leq \beta^\theta\beta = \beta^{\theta^+}$. And last, if γ is a nonzero limit ordinal,

$$\alpha^\gamma = \sup\{\,\alpha^\xi : \xi < \gamma\,\} \leq \sup\{\,\beta^\xi : \xi < \gamma\,\} = \beta^\gamma.$$

(iv) By its definition.

► B.7 **(iv′)** This follows from Lemma B.7, with a little fiddle to get around the single value (0^0) at which the exponential function fails to be order-preserving. The easiest fiddle is to define a function

$$f(\alpha,\beta) = \begin{cases} \alpha^\beta & \text{if } \alpha \neq 0 \ ; \\ \alpha^{\beta+1} & \text{if } \alpha = 0 \ . \end{cases}$$

and apply the lemma to that. ■

Most of the arithmetic properties are left as an amusing exercise. We will state and prove just one, which is interesting enough to rate a theorem all to itself.

C.9 Theorem: Base-α notation

Let α and β be ordinals with $\alpha \geq 2$. Then there is a natural number n and ordinals $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ and $\gamma_1, \gamma_2, \ldots, \gamma_n$ such that $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_n$, for each i, $0 < \gamma_i < \alpha$ and

$$\beta = \alpha^{\epsilon_n}\gamma_n + \alpha^{\epsilon_{n-1}}\gamma_{n-1} + \cdots + \alpha^{\epsilon_1}\gamma_1 \ .$$

(When $\beta = 0$, interpret this as meaning that $n = 0$, so that 0 is represented as an empty sum.)

Moreover, this representation is unique: given α and β, there is only one choice of n, ϵ_1, $\epsilon_2, \ldots, \epsilon_n$, and $\gamma_1, \gamma_2, \ldots, \gamma_n$ for which this equation holds.

Proof. Because of the special case of the statement when $\beta = 0$, we may assume now that $\beta \neq 0$. For the purposes of this proof, let us say that an expression $\alpha^{\epsilon_n}\gamma_n + \alpha^{\epsilon_{n-1}}\gamma_{n-1} + \cdots + \alpha^{\epsilon_1}\gamma_1$ is in "standard form" if n is a natural number, $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ are ordinals such that $\epsilon_1 < \epsilon_2 < \ldots, \epsilon_n$, and $\gamma_1, \gamma_2, \ldots, \gamma_n$ are ordinals such that $0 < \gamma_i < \alpha$ for each $i = 1, 2, \ldots, n$. Our aim is to prove that β can be represented uniquely by such a standard form. This is proved by induction over β.

► C.8 Since $\alpha \neq 0$, by (i) and (ii) of C.8 above, $\xi \mapsto \alpha^\xi$ is weakly order-preserving and continuous. Using the Intermediate Value Theorem, since $\alpha^0 \leq \beta < \alpha^\beta$ there exists an ordinal e such that $\alpha^e \leq \beta < \alpha^{e^+}$.

Now we use the division algorithm: there are ordinals g and ρ such that

$$\beta = \alpha^e g + \rho \qquad \text{and} \qquad \rho < \alpha^e.$$

Now if $\alpha > \beta$, we have $e = 0$, $g = \beta$ and $\rho = 0$, so we are finished: $\beta = \alpha^e g$ and set $n = 1$, $\epsilon_1 = e$ and $\gamma_1 = g$. Otherwise we have $\alpha \leq \beta$, so that $e \geq 1$ and $g \geq 1$. Then $\rho < \alpha^e < \beta$ and we can apply the inductive hypothesis: ρ can be expressed in standard form

$$\rho = \alpha^{\epsilon_n}\gamma_n + \alpha^{\epsilon_{n-1}}\gamma_{n-1} + \cdots + \alpha^{\epsilon_1}\gamma_1 \, .$$

Then, setting $\epsilon_{n+1} = e$ and $\gamma_{n+1} = g$, we have

$$\beta = \alpha^{\epsilon_{n+1}}\gamma_{n+1} + \alpha^{\epsilon_n}\gamma_n + \alpha^{\epsilon_{n-1}}\gamma_{n-1} + \cdots + \alpha^{\epsilon_1}\gamma_1$$

and, in order to show that this is also in standard form, it only remains to show that $\epsilon_{n+1} > \epsilon_n$, that is, that $e > \epsilon_n$. But, if $\epsilon_n \geq e$, we would have $\rho \geq \alpha^e$ so that $\beta = \alpha^e g + \rho \geq \alpha^e g + \alpha^e = g^+ \alpha^e$ contradicting the choice of g.

This establishes the existence of the representation for β.

Before proving uniqueness, we show that $\alpha^{\epsilon_n}\gamma_n + \alpha^{\epsilon_{n-1}}\gamma_{n-1} + \cdots + \alpha^{\epsilon_1}\gamma_1 < \gamma_n^+ \alpha^{\epsilon_n} \leq \alpha^{\epsilon_n^+}$ by induction over n. If $n = 1$, then $\alpha^{\epsilon_n}\gamma_n < \alpha^{\epsilon_n}\gamma_n^+ \leq \alpha^{\epsilon_n}\alpha = \alpha^{\epsilon_n^+}$ since $\gamma_n < \alpha$. Otherwise $n \geq 2$ and

$$\alpha^{\epsilon_n}\gamma_n + \alpha^{\epsilon_{n-1}}\gamma_{n-1} + \cdots + \alpha^{\epsilon_1}\gamma_1 < \alpha^{\epsilon_n}\gamma_n + \alpha^{\epsilon_{n-1}^+} \leq \alpha^{\epsilon_n}\gamma_n + \alpha^{\epsilon_n} = \alpha^{\epsilon_n}\gamma_n^+ \leq \alpha^{\epsilon_n}\alpha = \alpha^{\epsilon_n^+}.$$

To prove uniqueness, suppose that β has two standard representations

$$\begin{aligned}\beta &= \alpha^{\epsilon_n}\gamma_n + \alpha^{\epsilon_{n-1}}\gamma_{n-1} + \cdots + \alpha^{\epsilon_1}\gamma_1 \\ &= \alpha^{\epsilon'_m}\gamma'_m + \alpha^{\epsilon'_{m-1}}\gamma'_{m-1} + \cdots + \alpha^{\epsilon'_1}\gamma'_1\end{aligned}$$

Suppose that $\epsilon_n \neq \epsilon'_m$. Then wlog $\epsilon_n < \epsilon'_m$ and we have

$$\alpha^{\epsilon_n}\gamma_n + \alpha^{\epsilon_{n-1}}\gamma_{n-1} + \cdots + \alpha^{\epsilon_1}\gamma_1 < \alpha^{\epsilon_n^+} \leq \alpha^{\epsilon'_m} \leq \alpha^{\epsilon'_m}\gamma'_m + \alpha^{\epsilon'_{m-1}}\gamma'_{m-1} + \cdots + \alpha^{\epsilon'_1}\gamma'_1$$

contradicting the assumption. Therefore $\epsilon_n = \epsilon'_m$. Suppose now that $\gamma_n \neq \gamma'_m$. Then wlog $\gamma_n < \gamma'_m$ and

$$\alpha^{\epsilon_n}\gamma_n + \alpha^{\epsilon_{n-1}}\gamma_{n-1} + \cdots + \alpha^{\epsilon_1}\gamma_1 < \alpha^{\epsilon_n}\gamma_n^+ \leq \alpha^{\epsilon'_m}\gamma_m \leq \alpha^{\epsilon'_m}\gamma'_m + \alpha^{\epsilon'_{m-1}}\gamma'_{m-1} + \cdots + \alpha^{\epsilon'_1}\gamma'_1$$

again contradicting the assumption. We now know that $\epsilon_n = \epsilon_m$ and $\gamma_n = \gamma_m$. It follows that the leading terms of the two representations are the same. But then, so are their tails:

$$\alpha^{\epsilon_{n-1}}\gamma_{n-1} + \alpha^{\epsilon_{n-2}}\gamma_{n-2} + \cdots + \alpha^{\epsilon_1}\gamma_1 = \alpha^{\epsilon'_{m-1}}\gamma'_{m-1} + \alpha^{\epsilon'_{m-2}}\gamma'_{m-2} + \cdots + \alpha^{\epsilon'_1}\gamma'_1$$

and then, by induction, these tails are identical expressions also. ■

C.10 Definition: Disjoint unions

The disjoint union of two sets

In what follows, we will want to be able to manufacture the "disjoint union" of two sets, a set which is like the union $A \cup B$ except that $A \cap B = \varnothing$. Of course, if the given sets A and B we start with are not disjoint, we cannot simply use their union in this simple-minded way. What we do is create two identical copies of A and B, in such a way that they are guaranteed to be disjoint, and use the union of these copies. To do this we tag each element of A and B, using different tags for A and B. Convenient tags are 0 and 1, so we manufacture

$$\begin{aligned}\bar{A} &= \{\, \langle 0, a\rangle : a \in A \,\} \\ \bar{B} &= \{\, \langle 1, b\rangle : b \in B \,\}\end{aligned}$$

and now $\bar{A}$ and $\bar{B}$ are the required disjoint copies of A and B. Then define the *disjoint union* of A and B (which we will denote by $A \dot{\cup} B$) to be the set

$$A \dot{\cup} B = \bar{A} \cup \bar{B} = \{\, \langle 0, a\rangle : a \in A \,\} \cup \{\, \langle 1, b\rangle : b \in B \,\}.$$

The maps $A \to \bar{A}$ and $B \to \bar{B}$ given by $a \mapsto \langle 0, a\rangle$ and $b \mapsto \langle 1, b\rangle$ obviously map A and B one-to-one onto $\bar{A}$ and $\bar{B}$ respectively.

This construction becomes more interesting to us when the sets involved are ordered. In the definition above, let us now suppose that the sets A and B are ordered. We define an order on the disjoint union as follows:

Let $\langle i, a\rangle$ and $\langle j, b\rangle$ be members of the disjoint union $A \dot\cup B$. Then

$$\langle i, a\rangle \le \langle j, b\rangle \qquad \text{if and only if} \quad \text{(a)} \quad i < j \quad \text{or}$$
$$\text{(b)} \quad i = j \quad \text{and} \quad a \le b.$$

Note that the corresponding strict order is given by:

$$\langle i, a\rangle < \langle j, b\rangle \qquad \text{if and only if} \quad \text{(a)} \quad i < j \quad \text{or}$$
$$\text{(b)} \quad i = j \quad \text{and} \quad a < b.$$

The disjoint union of a family of sets

We extend the construction just discussed to any family (indexed collection) of sets.

Let $\langle A_i\rangle_{i\in I}$ be a family of sets indexed by the set I. Then the *disjoint union* of this family is the set

$$\dot\bigcup \langle A_i\rangle_{i\in I} \;=\; \bigcup \langle \bar{A}_i\rangle_{i\in I} \qquad \text{where} \qquad \bar{A}_i = \{\, \langle i, a\rangle : a \in A_i \,\} \quad \text{for each } i.$$

We have a *natural injection* $\eta_i : A_i \to \bar{A}_i$ given by $\eta_i(a) = \langle i, a\rangle$ for each i and (as is the point of this construction), $\bar{A}_i \cap \bar{A}_j = \varnothing$ for all $i \ne j$.

Let us now assume that the index set I and all the sets A_i are ordered. We define an order on the disjoint union as follows:

Let $\langle i, a\rangle$ and $\langle j, b\rangle$ be members of the disjoint union $\dot\bigcup \langle A_i\rangle_{i\in I}$. Then

$$\langle i, a\rangle \le \langle j, b\rangle \qquad \text{if and only if} \quad \text{(a)} \quad i < j \quad \text{or}$$
$$\text{(b)} \quad i = j \quad \text{and} \quad a \le b.$$

Note that the corresponding strict order is given by:

$$\langle i, a\rangle < \langle j, b\rangle \qquad \text{if and only if} \quad \text{(a)} \quad i < j \quad \text{or}$$
$$\text{(b)} \quad i = j \quad \text{and} \quad a < b.$$

C.11 Theorem: Properties of disjoint unions

Properties of the disjoint union of two sets

(i) If the given orders on A an B are both partial orders, then so is the new one defined on the disjoint union. If the given orders are both full orders, then so is the new one defined on the disjoint union. And, most importantly, if the given orders are both well orders, then so is the new one defined on the disjoint union.

(ii) $A \dot\cup B$ is the union of two disjoint sets, $\bar A = \{\langle 0, a\rangle : a \in A\}$ and $\bar B = \{\langle 1, b\rangle : b \in B\}$, which are order-isomorphic to A and B respectively, and every member of $\bar A$ is less than ($<$) every member of $\bar B$.

(iii) If A and B are order-isomorphic to A' and B' respectively, then $A \dot\cup B \simeq A' \dot\cup B'$.

(iv) (Converse of (ii)) If the ordered set X is the union of two disjoint sets, $\bar A$ and $\bar B$, which are order-isomorphic to A and B respectively, and every member of $\bar A$ is less than ($<$) every member of $\bar B$, then $X \simeq A \dot\cup B$.

Properties of the disjoint union of a family of sets

(i′) If the given orders on I and all the A_i are partial orders, then so is the new one defined on the disjoint union. If the given orders are all full orders, then so is the new one defined on the disjoint union. And, most importantly, if the given orders are all well orders, then so is the new one defined on the disjoint union.

(ii′) $\dot\bigcup\langle A_i\rangle_{i\in I}$ is the union of the sets $\bar A_i = \{\langle i, a\rangle : a \in A_i\}$ (for each $i \in I$), these are all disjoint (that is, $\bar A_i \cap \bar A_j = \varnothing$ whenever $i \neq j$), $\bar A_i \simeq A_i$ for every $i \in I$, and, if $i < j$, every member of $\bar A_i$ is less than ($<$) every member of $\bar A_j$.

(iii″) If $A_i \simeq A'_i$ for every $i \in I$, then $\dot\bigcup\langle A_i\rangle_{i\in I} \simeq \dot\bigcup\langle A'_i\rangle_{i\in I}$.

(iv′) (Converse of (ii)) If the ordered set X is the union of pairwise disjoint sets, $\bar A_i$ for all $i \in I$, which are order-isomorphic to A_i respectively, and, for every $i < j$, every member of $\bar A_i$ is less than ($<$) every member of $\bar A_j$, then $X \simeq \dot\bigcup\langle A_i\rangle_{i\in I}$.

Proof. These proofs can be left as a straightforward, if tedious, exercise. Clearly (i) – (iv) are all special cases of (i′) – (iv′) and so it is only necessary to prove the latter statements. ■

C.12 Remark

Given three sets, A, B and C, we can index them by $I = \{0, 1, 2\}$ and so define the disjoint union

$$A \dot\cup B \dot\cup C = \bar A \cup \bar B \cup \bar C = \{\langle 0, a\rangle : a \in A\} \ \cup \ \{\langle 1, b\rangle : b \in B\} \ \cup \ \{\langle 2, c\rangle : c \in C\} \ .$$

This is different from $A \dot\cup (B \dot\cup C)$ and $(A \dot\cup B) \dot\cup C$, since applying the definition above yields

$$\begin{aligned} A \dot\cup (B \dot\cup C) &= \{\langle 0, a\rangle : a \in A\} \ \cup \ \{\langle 1, \langle 0, b\rangle\rangle : b \in B\} \ \cup \ \{\langle 1, \langle 1, c\rangle\rangle : c \in C\} \\ (A \dot\cup B) \dot\cup C &= \{\langle 0, \langle 0, a\rangle\rangle : a \in A\} \ \cup \ \{\langle 0, \langle 1, b\rangle\rangle : b \in B\} \ \cup \ \{\langle 1, c\rangle : c \in C\} \ , \end{aligned}$$

however the natural mappings between these sets are order-isomorphisms, so we have an associative law (up to isomorphism) for the disjoint product:

$$(A \dot\cup B) \dot\cup C \simeq A \dot\cup (B \dot\cup C) \simeq A \dot\cup B \dot\cup C \ .$$

C.13 Theorem

(i) Let α and β be two ordinal numbers. Then $\alpha \dot\cup \beta \simeq \alpha + \beta$.

(ii) Let A and B be two well-ordered sets of order types α and β respectively. Then the disjoint union $A \dot\cup B$ is of order type $\alpha + \beta$.

Proof. First we observe that (ii) is an immediate corollary of (i), so it remains to prove (i), which we do by induction over β.

Firstly, $\alpha \dot\cup 0 = \alpha \dot\cup \varnothing = \alpha$, so the result is trivial in this case.

Next we observe that, for any ordinal α, $\alpha \dot\cup 1 \simeq \alpha^+$. (An order-isomorphism between these sets is easy to construct.)

Now suppose that β is a non-limit ordinal, $\beta = \theta^+$ say. Then (and here we make free use of A25(iii))

$$\begin{aligned}
\alpha \dot\cup \beta &= \alpha \dot\cup \theta^+ \\
&\simeq \alpha \dot\cup (\theta \dot\cup 1) && \text{just proved} \\
&\simeq (\alpha \dot\cup \theta) \dot\cup 1 && \text{by associativity of the disjoint union} \\
&\simeq (\alpha + \theta) \dot\cup 1 && \text{by the inductive hypothesis} \\
&\simeq (\alpha + \theta)^+ && \text{just proved} \\
&= \alpha + \theta^+ = \alpha + \beta
\end{aligned}$$

Finally, suppose that β is a nonzero limit ordinal. Observing that $\alpha + \beta$ is the union of α and the set $\{\alpha + \xi : \xi \in \beta\}$, that these subsets are disjoint and that every member of α is less that every member of $\{\alpha + \xi : \xi \in \beta\}$, we conclude that $\alpha + \beta \simeq \alpha \dot\cup \{\alpha + \xi : \xi \in \beta\}$. Also, the map $\beta \to \{\alpha + \xi : \xi \in \beta\}$ defined by $\xi \mapsto \alpha + \xi$ is an order-isomorphism, so $\beta \simeq \{\alpha + \xi : \xi \in \beta\}$ and therefore $\alpha + \beta \simeq \alpha \dot\cup \beta$. ■

C.14 Definition: Ordering a product

The *lexicographic order from the right* on the cartesian product $A \times B$ of two ordered sets is defined by

$$\begin{aligned}
\langle a, b\rangle \le \langle a', b'\rangle \quad \Leftrightarrow \quad & \text{either} \quad b < b' \\
& \text{or} \quad b = b' \quad \text{and} \quad a \le a'.
\end{aligned}$$

Note that the corresponding strict order is given by

$$\begin{aligned}
\langle a, b\rangle < \langle a', b'\rangle \quad \Leftrightarrow \quad & \text{either} \quad b < b' \\
& \text{or} \quad b = b' \quad \text{and} \quad a < a'.
\end{aligned}$$

From now on, when we refer to the product $A \times B$ of two ordered sets, we will assume that it has been endowed with this order.

(One can also define the *lexicographic order from the left*, however this order turns out not to be useful in the context of transfinite arithmetic.)

The cartesian product of a finite number of sets, $A_0 \times A_1 \times \cdots \times A_{n-1}$ can be ordered lexicographically from the right in the same way. It is easiest to define the strict order, the order being

$$\langle a_0, a_1, \ldots, a_{n-1}\rangle < \langle b_0, b_1, \ldots, b_{n-1}\rangle \quad \Leftrightarrow \quad \text{there is some } k < n \text{ such that } a_k < b_k \text{ and } a_i = b_i \text{ for all } i > k.$$

As before, it is possible to order lexicographically from the left, but we will not be interested in this here. If you attempt to extend this definition to order a cartesian product with an infinite number of factors, the definition makes sense, but the resulting relation has problems. (It is an interesting exercise to see what goes wrong: assume the index set I and all the A_i are well-ordered, and see what can be said about the product, when ordered lexicographically either from the left and from the right.)

C.15 Theorem: Properties of the ordered product

(i) If A and B are well-ordered sets, then so is $A \times B$.

(And, as usual, if A and B are both partially ordered or both fully ordered, then $A \times B$ is partially or fully ordered respecively too.)

(ii) If A, B, A' and B' are ordered sets and $A \simeq A'$ and $B \simeq B'$ then $A \times B \simeq A' \times B'$.

(iii) A sort of associative law: if A, B and C are ordered sets, then $(A \times B) \times C \simeq A \times (B \times C) \simeq A \times B \times C$.

(iv) A sort of distributive law: if A, B and C are ordered sets, then $A \times (B \dot\cup C) \simeq (A \times B) \dot\cup (A \times C)$.

(v) If A is an ordered set and $\{p\}$ is a singleton, then $A \times \{p\} \simeq A$.

Proof. Another interesting exercise. ■

C.16 Theorem

(i) Let α and β be two ordinal numbers. Then $\alpha \times \beta \simeq \alpha\beta$.

(ii) Let A and B be two well-ordered sets of order types α and β respectively. Then $A \times B$ is of order type $\alpha\beta$.

Proof. (ii) is an immediate corollary of (i), so it is enough to prove (i).

We define a function $f : \alpha \times \beta \to \alpha\beta$ by $f(\xi, \eta) = \alpha\eta + \xi$. We will show that this is an order-isomorphism.

First we must check that its range is within $\alpha\beta$ as claimed. So, for $\langle \xi, \eta\rangle \in \alpha \times \beta$ we have $\xi < \alpha$ and $\eta < \beta$ and thus

$$f(\xi, \eta) \;=\; \alpha\eta + \xi \;<\; \alpha\eta + \alpha \;=\; \alpha\eta^+ \;\leq \alpha\beta$$

so $f(\xi,\eta) \in \alpha\beta$ as required.

To see that f is surjective, suppose that θ is any member of $\alpha\beta$, that is, $\theta < \alpha\beta$. By the division algorithm, there is ξ, η such that $\theta = \alpha\eta + \xi$ with $\xi < \alpha$. Then also $\eta < \beta$, because otherwise we would have $\eta \geq \beta$ and then $\alpha\eta + \xi \geq \alpha\eta \geq \alpha\beta$. Thus we have $\xi < \alpha$ and $\eta < \beta$ such that $f(\xi,\eta) = \alpha\eta + \xi = \theta$. This shows that f maps α onto $\alpha\beta$.

It remains to show that f is strictly increasing. So suppose that $\langle \xi, \eta\rangle < \langle \xi', \eta'\rangle$. The either $\eta < \eta'$ or else $\eta = \eta'$ and $\xi < \xi'$.

In the first case, $\eta^+ \leq \eta'$ so

$$f(\xi,\eta) \;=\; \alpha\eta + \xi \;<\; \alpha\eta + \alpha \;=\; \alpha\eta^+ \;\leq\; \alpha\eta' \;\leq \alpha\eta' + \xi' \;=\; f(\xi',\eta')$$

and in the second case $f(\xi,\eta) \;=\; \alpha\eta + \xi \;<\; \alpha\eta + \xi' \;=\; f(\xi',\eta')$. ■

C.17 Remarks

Notice the similarities between the two definitions of $A \dot\cup B$ and $A \times B$. In the case $A \dot\cup A$ we have

$$A \dot\cup A = \{0,1\} \times A = 2 \times A$$

and in general

$$A \dot\cup A \dot\cup \ldots \dot\cup A \text{ (n copies)} = \{0,1,\ldots,n-1\} \times A = n \times A$$

which is another way of establishing that $\alpha + \alpha = 2\alpha$, $\alpha + \alpha + \alpha = 3\alpha$ and so on.

Let $\langle\beta\rangle_{i<\alpha}$ be a sequence of ordinals, indexed by the ordinal α. We define the sum $\sum_{i<\alpha}\beta_i$ as follows:

$$\sum_{i<0}\beta_i = 0 \qquad , \qquad \sum_{i<\alpha^+}\beta_i = \sum_{i<\alpha}\beta_i + \beta_{\alpha^+}$$

and, for a limit ordinal λ, $\displaystyle\sum_{i<\lambda}\beta_i = \sup\{\sum_{i<\xi}\beta_i : \xi < \lambda\}.$

It is easy to prove that, in the case where all the α_i are the same, $\beta_i = \beta$ say,

$$\sum_{i<\alpha}\beta = \alpha\beta.$$

C.18 Ordering functions

Let A and B be two well-ordered sets and, for the sake of this definition, let us write $\underline{0}$ for the least member of A. If f is a function $B \to A$, then its *support* is the set

$$\text{spt}\, f = \{y : y \in B \;,\; f(y) \neq \underline{0}\}.$$

Then $A^{[B]}$ denotes the set of all functions $f : B \to A$ of "finite support", that is, for which spt f is a finite set.

This set is ordered thus:

$$f < g \quad \Leftrightarrow \quad \text{there is } b \in B \text{ such that } \quad f(b) < g(b) \\ \text{and} \quad f(y) = g(y) \text{ for all } y > b.$$

C.19 Theorem

(i) If A and B are well-ordered sets, then so is $A^{[B]}$.

(ii) If A, B, A' and B' are well-ordered sets and $A \simeq A'$ and $B \simeq B'$, then $A^{[B]} \simeq A'^{[B']}$.

(iii) $A^{[\varnothing]}$ is a singleton set.

(iv) Let $\{p\}$ be a singleton set. Then $A^{[\{p\}]} \simeq A$

(v) If A, B and C are well-ordered sets, then $A^{[B \dot\cup C]} \simeq A^{[B]} \times A^{[C]}$.

Proof. (i) Suppose that $f \leq g$ and $g \leq f$. We want to conclude that $f = g$, so let us assume that $f \neq g$ and deduce a contradiction. We now have $f < g$ and $g < f$, and so

$$\text{there is some } b \in B \text{ such that } \quad f(b) < g(b) \quad \text{and} \quad f(y) = g(y) \text{ for all } y > b. \quad (6.1)$$

$$\text{and there is some } b' \in B \text{ such that } \quad f(b') > g(b') \quad \text{and} \quad f(y) = g(y) \text{ for all } y > b'. \quad (6.2)$$

If $b = b'$, then $f(b) < g(b)$ from (1) and $f(b) > g(b)$ from (2), which is impossible. Therefore $b \neq b'$, and we may assume wlog that $b < b'$. But then $f(b') = g(b')$ from (1) and $f(b') > g(b')$ from (2), another contradiction.

Now we show that every nonempty subset of $A^{[B]}$ has a least element. We work by induction over B; the inductive hypothesis is: if J is a proper initial segment of B, then $A^{[J]}$ is well-ordered.

Let F be a nonempty subset of $A^{[B]}$. Choose a function $f \in F$ and let m be the maximum member of the (finite!) support of f. Then we have

$$f(y) = \underline{0} \qquad \text{for every } y > m.$$

Let us now write G for the set of all such functions:

$$G = \{g : g \in F \ , \ g(y) = \underline{0} \text{ for all } y > m\}.$$

Because of our choice of f and m, the set G is nonempty. Therefore the set $U = \{g(m) : g \in G\}$ is also nonempty. Now U is a subset of the well-ordered set A and so has a least element, u say: $u = \min U$. This means that there is a function h such that $h \in G$ and $h(m) = u$. Let V be the set of all such functions:

$$V = \{g : g \in G \text{ and } g(m) = u\}.$$

We note that $h \in V$ and so V is nonempty.

We will now show that, if $f \in F$ and $f \notin V$, then $f > h$. If $f \in F \setminus V$, then either

(a) $f \notin G$, in which case $f(y) \neq \underline{0}$ for some $y > m$, in which case there is a greates such y (since spt f is finite). Then $f(y) > \underline{0} = h(y)$ and $f(y') = \underline{0} = h(y')$ for all $y' > y$. Thus $f > h$.

or (b) $f \in G \setminus V$, in which case $f(m) \neq u$. By choice of u, $f(m) > u$ and, since $f \in G$, $f(y) = \underline{0} = h(y)$ for all $y > m$. Again $f > h$.

It is now sufficient to show that V has a least element.

Let $I = I(m) = \{x : x \in B, x < m\}$. Consider the map $f \mapsto f|_I$ (the restriction of f to I) which is a function $A^{[B]} \to A^{[I]}$. We show that the restriction of this map to V is strictly order-preserving and therefore an order-isomorphism with its range which, being a nonempty subset of $A^{[I]}$, has a least element, by the inductive hypothesis. That will complete the proof.

Let $f, g \in V$, $f < g$. Then there is some $b \in B$ such that $f(b) < g(b)$ and $f(y) = g(y)$ for all $y > b$. Noting that $f(y) = g(y)$ for all $y \geq m$, we see that $b < m$. Then $f|_I(b)$ and $g|_I(b)$ are defined and we have

$$f|_I(b) = f(b) < g(b) = g|_I(b)$$
$$\text{and} \qquad f|_I(y) = f(y) = g(y) = g|_I(y) \qquad \text{for all } y \text{ such that } b < y < m.$$

Then $f|_I < g|_I$ as required.

(ii) The proof of this is straightforward.

(iii) There is only one function $\varnothing \to A$, namely the empty function, and it has finite support.

(iv) It is clear that the mapping $A^{[\{p\}]}$ defined by $f \mapsto f(p)$ is an order-isomorphism.

(v) Given a function $f : B \dot\cup C \to A$, define two functions $f_1 : B \to A$ and $f_2 : C \to A$ by

$$f_1(b) = f(0, b) \qquad \text{and} \qquad f_2(c) = f(1, c).$$

Then it is easy to check that $f \mapsto \langle f_1, f_2 \rangle$ defines an order-isomorphism $A^{[B \dot\cup C]} \to A^{[B]} \times A^{[C]}$. ■

C.20 Theorem

(i) Let α and β be two ordinal numbers. Then $\alpha^{[\beta]} \simeq \alpha^\beta$.

(ii) Let A and B be well-ordered sets of order types α and β respectively. Then $A^{[B]}$ is of order type α^β.

Proof. (ii) is an immediate corollary of (i), so we only prove (i).

We define a function $\Phi : \alpha^{[\beta]} \to \alpha^\beta$ as follows: for any $f \in \alpha^{[\beta]}$ (function $f : \beta \to \alpha$ of finite support $S = \{\sigma_1 < \sigma_2 < \cdots < \sigma_n\}$), $\Phi(f)$ is the ordinal

$$\Phi(f) = f(\sigma_n)\alpha^{\sigma_n} + f(\sigma_{n-1})\alpha^{\sigma_{n-1}} + \cdots + f(\sigma_1)\alpha^{\sigma_1}.$$

► C.9 That Φ is an order-isomorphism follows from Theorem C.9. ■

D Cardinality

In this section we will consider the problem of comparing the *sizes* of infinite sets. The crucial ideas are:—

D.1 Injective, surjective and bijective functions

Some synonyms (for a function $f : A \to B$):

(1) f is an injection, f is injective, f is mono, f is one-to-one.

(2) f is a surjection, f is surjective, f is epi, f is onto B.

(3) f is a bijection, f is bijective, f is iso, f is one-to-one and onto B.

An injection is defined to be a function $f : A \to B$ such that, for all $a_1, a_2 \in A$, $\quad f(a_1) = f(a_2) \;\Rightarrow\; a_1 = a_2$.

A surjection is defined to be a function $f : A \to B$ such that, for all $b \in B$, there exists some $a \in A$ such that $f(a) = b$.

A bijection is defined to be a function which is both an injection and a surjection.

Appeal to these definitions is probably the most common way of proving that a function has these properties.

An alternative way of proving that a function is a bijection is to appeal to the fact that a function f is a bijection if and only if it has an inverse, that is, a function $g : B \to A$ such that $g(f(a)) = a$ for all $a \in A$ and $f(g(b)) = b$ for all $b \in B$. In this case the inverse is usually denoted f^{-1} and is unique.

It is useful to know that the composite of two injections is an injection, of two surjections is a surjection and of two bijections is a bijection. Also that the identity function on any set is a bijection, and therefore also an injection and a surjection. (Proofs are straightforward.)

D.2 Comparison of size

We say that the sets A and B are *the same size*, denoted $A \approx B$, if there is a bijection $A \to B$.

(Synonyms are: A is *equipollent* to B, A has the same *cardinality* as B.)

We say that set A is *no larger* than set B, denoted $A \preccurlyeq B$, if there is an injection $A \to B$.

(Synonyms are: B is *no smaller* than A, A is *dominated* by B, A has *cardinality not exceeding* that of B.)

We say that set A is *smaller* than set B, denoted $A \prec B$, if $A \preccurlyeq B$ and $A \not\approx B$.

(Synonyms are: B is *larger* than A, B *strictly dominates* A, A has *cardinality less* than that of B.)

D.3 Theorem

Being the same size (equipollence) is an equivalence relation on the class of all sets. That is, this relation satisfies

(i) $A \approx A$ for all sets A.

(ii) If $A \approx B$ then $B \approx A$ for all sets A and B.

(iii) If $A \approx B$ and $B \approx C$ then $A \approx C$ for all sets A, B and C.

Proof. (i) The identity function $A \to A$ is a bijection.

(ii) If $A \approx B$ there is a bijection $f : A \to B$ and then $f^{-1} : B \to A$ is a bijection $B \to A$.

(iii) If $A \approx B$ and $B \approx C$ there are bijections $f : A \to B$ and $g : B \to C$ and then $g \circ f$ is a bijection $A \to C$. ■

D.4 Examples

(i) $X \approx \varnothing$ if and only if $X = \varnothing$.

(ii) Normal counting of finite sets: the set A has n members if and only if $A \approx n$. (This probably should be the definition.)

(iii) For $m, n \in \mathbb{N}$, $m \approx n$ if and only if $m = n$.

(iv) A set A is finite if and only if there is some natural number n such that $A \approx n$. (This is a definition.)

(v) Consider the set $\mathbb{N} \setminus \{0\}$. The function $x \mapsto x + 1$ is a bijection $\mathbb{N} \to \mathbb{N} \setminus \{0\}$, so these sets are the same size. Thus it is perfectly possible for a set to be the same size as one of its proper subsets.

(vi) A set is infinite (that is, not finite in the sense defined above) if and only if it is the same size as one of its proper subsets.

(vii) Let E be the set of even natural numbers, $E = \{0, 2, 4, 6, \ldots\}$. Then $E \approx \mathbb{N}$.

(viii) Any two nonempty open intervals (a, b) and (c, d) in $\mathbb{R}$ are the same size. (Use the linear function which takes $a \to c$ and $b \to d$.)

(ix) Any nonempty open interval (a, b) in $\mathbb{R}$ is the same size as $\mathbb{R}$ itself. (To see that $\mathbb{R} \approx (-1, +1)$, use the function $x \mapsto x/\sqrt{1 + x^2}$.)

(x) The open interval $(-1, +1)$ is the same size as the closed interval $[-1, +1]$.

(xi) If $A \approx A'$ and $B \approx B'$ then $A \dot\cup B \approx A' \dot\cup B'$, $A \times B \approx A' \times B'$, $B^A \approx B'^{A'}$, $\mathcal{P}(A) \approx \mathcal{P}(A')$ and so on. (Note that $A \setminus B$ need not be the same size as $A' \setminus B'$, even if $B \subseteq A$ and $B' \subseteq A'$. What about $A \cup B$?)

Many examples are much more easily proved using the Schröder-Bernstein Theorem (see below). Example (x) above is the first such.

D.5 Definition

A set is *countable* if it is either finite or $\approx \mathbb{N}$. It is *countably infinite* if it is $\approx \mathbb{N}$. (*Denumerable* is a common synonym for countably infinite.)

D.6 Theorem

For any set X, $\quad X \not\approx \mathcal{P}(X)$.

Proof. Suppose otherwise. Then there is a set X and a bijection $F : X \to \mathcal{P}(X)$. Let R be the subset of X defined by $R = \{\, x : x \in X \text{ and } x \notin F(x) \,\}$. Then $R \subseteq X$, so there is some $r \in R$ such that $R = F(r)$. Now we ask: does $r \in R$ or not?

$$r \in R \quad \Rightarrow r \in F(r) \quad \Rightarrow \quad r \notin R \qquad \text{which proves that } r \notin R;$$

$$r \notin R \quad \Rightarrow r \notin F(r) \quad \Rightarrow \quad r \in R \qquad \text{which proves that } r \in R;$$

This is a contradiction. ■

D.7 Theorem

For any set A, $\quad \mathcal{P}(A) \approx 2^A$.

(Here 2^A is the set of all functions $A \to 2$, that is, all functions $A \to \{0, 1\}$.

Proof. For each subset X of A we define its *characteristic function* χ_X by

$$\chi_X(x) = \begin{cases} 1 & \text{if } x \in X \ ; \\ 0 & \text{if } x \notin X \ . \end{cases}$$

It is easy to see that the function $X \mapsto \chi_X$ is a bijection $\mathcal{P}(X) \to 2^X$. ■

D.8 Theorem

The relation $\preccurlyeq$ is a full preorder and $\approx$ is the corresponding equivalence relation. That is, this relation satisfies

(i) $A \preccurlyeq A$ for all sets A.

(ii) If $A \preccurlyeq B$ and $B \preccurlyeq A$ then $A \approx B$ for all sets A and B.

(iii) If $A \preccurlyeq B$ and $B \preccurlyeq C$ then $A \preccurlyeq C$ for all sets A, B and C.

(iv) AC For any two sets A and B, either $A \preccurlyeq B$ or $B \preccurlyeq A$.

Proof. (i) The identity function on A is an injection.

(ii) is the Schröder-Bernstein Theorem, proved below.

(iii) If $A \preccurlyeq B$ and $B \preccurlyeq C$ then there are injections $f : A \to B$ and $g : B \to C$. But then $g \circ f$ is an injection $A \to C$.

(iv) We invoke the Axiom of Choice in its avatar as the Well Ordering Principle, and assume that both A and B can be well ordered. Then, by D11, either there is an order-isomorphism from A to an initial segment of B, or there is an order-isomorphism from B to an initial segment of A. But such an order isomorphism is injective.

■

D.9 The Schröder-Bernstein Theorem

If $A \preccurlyeq B$ and $B \preccurlyeq A$ then $A \approx B$.

Proof. From the assumptions, there are injections $f : A \to B$ and $g : B \to A$.

Consider a member b of B. There is no assumption that the function f is surjective, so there may or may not be an element a of A such that $f(a) = b$. If there is such an a, then there is only one (since f is injective), and we will call it the *parent* of b. If there is no such a, we will say that b is an *orphan*. Let B_0 be the set of orphans in B, that is,

$$B_0 = \{\, b : b \in B \text{ and there is no } a \in A \text{ such that } (f(a) = b) \,\}.$$

Now let a be a member of A. In the same way, we will say that a member b of B is the *parent* of a if $g(b) = a$. If a has a parent, then it has only one. Otherwise a is an *orphan* in A. The set of all orphans in A will be denoted by A_0.

Now consider a member a of A, and trace its ancestry back as far as possible. It may have a parent in B, which in turn may have a parent in A, which in turn ... and so on. One of three things must happen:

- Following ancestry back must terminate with an orphan in A_0. We will say that such elements are "descended from A".
- Following ancestry back must terminate with an orphan in B_0. We will say that such elements are "descended from B".
- Following ancestry back never terminates at all. We will say that such elements are "infinitely descended".

Because the functions f and g are injective, it is easy to see that these three possibilities are mutually exclusive. Therefore we can define three subsets of A:

$$A_A = \text{members of } A \text{ which are descended from A,}$$

$$A_B = \text{members of } A \text{ which are descended from B, and}$$

$$A_\infty = \text{members of } A \text{ which are infinitely descended.}$$

Every member of A is a member of exactly one of these subsets. Note also that $A_0 \subseteq A_A$ (tracing back the ancestry of an orphan stops with itself).

We can also trace back the ancestry of members of B, thus dividing it up into three corresponding subsets:

B_A = members of B which are descended from A,

B_B = members of B which are descended from B, and

B_∞ = members of B which are infinitely descended.

Again, every member of B is a member of exactly one of these subsets, and $B_0 \subseteq B_B$.

Now consider the restrictions f_A, f_B and f_∞ of the function f to the sets A_A, A_B and A_∞ respectively. Since f is injective, so are these three restrictions. By the definitions of B_A, B_B and B_∞, we see that

- f_A is onto B_A, and so is a bijection $A_A \to B_A$.
- f_∞ is onto B_∞, and so is a bijection $A_\infty \to B_\infty$.
- f_B is probably not onto B_B — it is only onto $B_B \setminus B_0$.

However, by a similar argument, the restriction g_B of g to B_B is a bijection $B_B \to A_B$, and so it has an inverse $g_B^{-1} : A_B \to B_B$, which must be a bijection also. We can now put the last three bijections together and define $\varphi : A \to B$ by

$$\varphi(a) \;= \begin{cases} f_A(a) & \text{if } a \in A_A \ ; \\ g_B^{-1}(a) & \text{if } a \in A_B \ . \end{cases}$$

It follows from the observations above that φ is a bijection $A \to B$. ■

D.10 Theorem AC

Let A and B be sets, $A \neq \varnothing$. Then $A \preccurlyeq B$ if and only if there is a surjection $B \to A$.

(The proof that the existence of a surjection implies $A \preccurlyeq B$ requires the Axiom of Choice; the converse does not.)

Proof. Suppose first that $A \preccurlyeq B$. Then there is an injection $f : A \to B$. Also, since A is nonempty, we can choose a particular element, a_0 say, of A. Now we can construct a surjection $B \to A$ as follows. For any b in the range of f, let $g(b)$ be the unique $a \in A$ for which $f(a) = b$. For any b not in the range of f, define $g(b) = a_0$. It is easy to see that then g is a surjection as claimed.

Now suppose that there exists a surjection $g : B \to A$. Consider the sets

$$g^{-1}(a) = \{\, b : b \in B \text{ and } g(b) = a \,\},$$

defined for each $a \in A$. These sets form a partition of B, so they are disjoint and all nonempty. Invoking the Axiom of Choice, there is a choice function $c : \mathcal{P}(B) \setminus \varnothing \to B$ such that $c(X) \in X$ for every nonempty subset of B. Define $f(a) = c(g^{-1}(a))$ for all $a \in A$. This defines a function $A \to B$, and, since all the sets $g^{-1}(a)$ are disjoint, it is an injection. ■

D.11 Examples

(i) A set is countable if and only if it is $\preccurlyeq \mathbb{N}$.

(ii) Every infinite set has a countably infinite subset.

(iii) If $A \subseteq B$ then $A \preccurlyeq B$.

(iv) $A \prec \mathcal{P}(A)$ for any set A.

There is an obvious injection $A \to \mathcal{P}(A)$, namely $a \mapsto \{a\}$, so $A \preccurlyeq \mathcal{P}(A)$. We have already proved that $A \not\approx \mathcal{P}(A)$.

(v) It is now easy to prove that the two intervals $[-1,+1]$ and $(-1,+1)$ in $\mathbb{R}$ are the same size. Firstly $(-1,+1) \subseteq [-1,+1]$ and so $(-1,+1) \preccurlyeq [-1,+1]$. Also, it is easy to show that $[-1,+1] \approx [-\frac{1}{2},+\frac{1}{2}]$ and $[-\frac{1}{2},+\frac{1}{2}] \subseteq (-1,+1)$.

(vi) By a similar argument, any nontrivial interval in $\mathbb{R}$, open, closed, half-open, finite or infinite (in length) is the same size as $\mathbb{R}$. (By "nontrivial" here I mean nonempty and not a singleton interval $[a,a]$).

(vii) $2^{\mathbb{N}} \approx \mathbb{R}$.

(Consequently $\mathbb{R}$ is uncountable. By previous examples then, neither is any nontrivial interval in $\mathbb{R}$.)

It is easiest to show that $2^{\mathbb{N}}$ is the same size as the interval $[0,1)$ of reals. The trick is to represent the reals in this interval as an infinite binary expansion. If we write $x = 0.d_0d_1d_2\ldots$ in this notation, we mean

$$x \;=\; \frac{d_0}{2} + \frac{d_1}{2^2} + \frac{d_2}{2^3} + \cdots$$

where each d_i is either 0 or 1. If we disallow sequences of digits which terminate with an infinite number of 1's, the representation is unique. Thus we have an injection from $[0,1)$ into the set of all sequences in $\{0,1\}$, indexed by $\mathbb{N}$, that is, $2^{\mathbb{N}}$. This proves that $R \preccurlyeq 2^{\mathbb{N}}$.

To prove the opposite relation, observe that, to every such sequence of 0's and 1's, there is a real number x in $[0,1)$ defined by

$$x \;=\; \frac{d_0}{3} + \frac{d_1}{3^2} + \frac{d_2}{3^3} + \cdots$$

and now, because the denominators are powers of 3, distinct sequences define distinct reals; therefore this defines an injection from $2^{\mathbb{N}}$ into $[0,1)$, proving that $2^{\mathbb{N}} \preccurlyeq [0,1)$.

(viii) (Corollary) $\mathbb{N} \prec \mathbb{R}$.

(ix) $\mathbb{N}^2 \approx \mathbb{N}$ (and so $\mathbb{N}^2$ is countably infinite).

There is an obvious injection $\mathbb{N} \to \mathbb{N}^2$ given by $n \mapsto \langle 0, n\rangle$. There is also an injection $\mathbb{N}^2 \to \mathbb{N}$ given by $\langle m, n\rangle \mapsto 2^m 3^n$.

(x) $\mathbb{R}^2 \approx \mathbb{R}$.

From $[0,1) \approx \mathbb{R}$ it is easy to prove that $[0,1) \times [0,1) \approx \mathbb{R} \times \mathbb{R}$, that is $[0,1)^2 \approx \mathbb{R}^2$. It is now sufficient to show that $[0,1)^2 \approx [0,1)$. That $[0,1) \preccurlyeq [0.1)^2$ is obvious, so we show that $[0,1)^2 \preccurlyeq [0,1)$.

Given a pair $\langle x, y\rangle$ in $[0,1)^2$, write each as its proper decimal expansion

$$x = 0.d_1d_2d_3\ldots$$

$$y = 0.e_1e_2e_3\ldots$$

Here "proper" means we never use an expansion which ends with an infinite sequence of 9's. Now define z by alternating digits in these expansions

$$z = 0.d_1e_1d_2e_2d_3e_3\ldots.$$

Then the function $\langle x, y\rangle \mapsto z$ (where z is defined as above) defines an injection $[0,1)^2 \to [0,1)$.

(xi) $\mathbb{Z}$ is countably infinite.

Define a bijection $\mathbb{Z} \to \mathbb{N}$ by

$$f(n) = \begin{cases} 2n & \text{if } n \geq 0\ ; \\ -2n-1 & \text{if } n < 0\ . \end{cases}$$

Some more examples as exercises

(xii) A set A is infinite if and only if $\mathbb{N} \preccurlyeq A$.

(xiii) $\mathbb{N}^{\mathbb{N}} \approx 2^{\mathbb{N}}$

(xiv) $\mathbb{R}^{\mathbb{N}} \approx \mathbb{R}$

(xv) The set of all *finite* sequences of natural numbers is countably infinite.

(xvi) The set of all functions $\mathbb{R} \to \mathbb{R}$ is $\mathbb{R}^{\mathbb{R}}$ and therefore $\succ \mathbb{R}$. What about the set of all *continuous* functions $\mathbb{R} \to \mathbb{R}$?

D.12 The Continuum Hypothesis

We know (Example (i) above) that there is no infinite set which is smaller than $\mathbb{N}$. We also know that $\mathbb{N}$ is smaller than $\mathbb{R}$. The question arises: is there any set of intermediate size here, that is, any set X such that $\mathbb{N} \prec X \prec \mathbb{R}$?

The Continuum Hypothesis states that there is no such set.

As its name implies, the Continuum Hypothesis is not a theorem of **VBG**. It has the same status as the Axiom of Choice: it is independent of **VBG**, in the sense that (writing CH for this Hypothesis) both **VBG**+CH and **VBG**+$\neg$CH are consistent theories (if **VBG** itself is consistent).

Given the fact that the Axiom of Choice is required for a number of useful results in this area, one might wonder if it could be used to prove the Continuum Hypothesis or its negation. But

no, the Continuum Hypothesis remains independent, even in the presence of the Axiom of Choice. More precisely, both **VBG**+AC+CH and **VBG**+AC+$\neg$CH are consistent (provided **VBG** is).

The *Generalised Continuum Hypothesis* is that, for any infinite set X, there is no set Y such that $X \prec Y \prec 2^X$. It has the same independence status as the ordinary one.

E Cardinal numbers

One can use the natural numbers to measure the size of finite sets, in the sense that every finite set is the same size as exactly one natural number. It would be nice if there was some set of canonical "numbers" which we could use to measure the size of infinite sets in the same way.

If we are prepared to accept the Axiom of Choice then this can be done, for then every set X can be well ordered and so made order-isomorphic to an ordinal number, its order type. But then, since an order-isomorphism is a bijection, it is the same size as its order type. Now forget the well ordering: we are left knowing that every set X is the same size as at least one ordinal number. Define its *cardinal number* or *cardinality* as the *least* ordinal number which is the same size as it.

$$\#X = \min\{\, \alpha : \alpha \text{ is an ordinal and } X \approx \alpha \,\} \ .$$

It is not difficult now to see that the ordinal numbers which can turn up in this role are just those ordinals which are not the same size as any of their predecessors. Now let us define all this in a more logical order.

E.1 Definition AC

A *cardinal number* is an ordinal number which is not the same size as any of its predecessors. In other words, it is an ordinal number α with the property that

$$\text{if} \qquad \xi < \alpha \qquad \text{then} \qquad \xi \not\approx \alpha \ .$$

Since, if $\xi < \alpha$, we have $\xi \subseteq \alpha$, this condition can be rewritten

$$\text{if} \qquad \xi < \alpha \qquad \text{then} \qquad \xi \prec \alpha \ .$$

Now let X be any set. We define its *cardinality* to be the least ordinal number which is the same size as X:

$$\#X = \min\{\, \alpha : \alpha \text{ is an ordinal and } X \approx \alpha \,\} \ ,$$

which is, of course, the unique cardinal number which is the same size as X. This is also called the *cardinal number* of X, or even the *size* of X.

(The definition of a cardinal number does not require the Axiom of Choice. The definition of the cardinality of a set requires the Axiom of Choice in its tacit assumption that every set does have a cardinality. Should we take this definition to mean that we are defining cardinality for sets which happen to be the same size as some ordinal number, accepting the possibility that some sets may not qualify, then the Axiom of Choice is not involved.)

E.2 Examples

(i) No natural number is the same size as any of its predecessors. Therefore every natural number is a cardinal number.

(ii) The ordinal $\omega = \mathbb{N}$ is infinite and therefore not the same size as any of its predecessors, which are the natural numbers. Therefore ω is the first infinite cardinal number, and in this context it is traditionally denoted $\aleph_0$ (read "aleph-null").

(iii) The ordinals $\omega+1, \omega+2, \ldots, 2\omega, \ldots, 3\omega, \ldots, \omega^2, \ldots$ are all countable. Therefore they are all the same size as ω, which means that none of them are cardinal numbers.

(iv) The cardinality of $\mathbb{R}$ is the same as that of the set of all functions $\aleph_0 \to 2$ and so is denoted $2^{\aleph_0}$ (more on this notation below).

(v) The Continuum Hypothesis is that there is no cardinal number between $\aleph_0$ and $2^{\aleph_0}$.

(vi) $A \approx B$ if and only if $\#A = \#B$,

$A \preccurlyeq B$ if and only if $\#A \leq \#B$

and $A \prec B$ if and only if $\#A < \#B$.

E.3 Cardinal arithmetic

As with ordinal numbers, we can define and investigate an arithmetic of cardinal numbers. For any two cardinal numbers α and β, we can define $\alpha + \beta$, $\alpha\beta$ and β^α as the cardinalities of sets $A \dot\cup B$, $A \times B$ and B^A, where A and B are sets of cardinality α and β respectively.

We must be careful here because these definitions of addition and multiplication are different from those made for ordinals in Section C. We must also be careful because we now have three definitions of the notation β^α: the definition for ordinal numbers in Definition C.7, the definition for cardianals just given and also, since α and β are sets, β^α has been defined as the set of all functions $\alpha \to \beta$. We will not be using the ordinal definition any more in these notes. To save confusion (I hope), in this section I will use the notation $\mathcal{F}(A \to B)$ for the set of all functions $A \to B$.

► C
► C.7

The definitions, written properly, are: let α and β be cardinals. Then

$\alpha + \beta$ is the cardinality of the set $\alpha \dot\cup \beta$, that is, $\alpha + \beta = \#(\alpha \dot\cup \beta)$.

$\alpha\beta$ is the cardinality of the set $\alpha \times \beta$, that is, $\alpha\beta = \#(\alpha \times \beta)$.

β^α is the cardinality of the set $\mathcal{F}(\alpha \to \beta)$, that is, $\beta^\alpha = \#\mathcal{F}(\alpha \to \beta)$.

As an immediate corollary of Example D.4(xi), we have: If A and B are sets of cardinalities α and β respectively, then

► D.4

$$\#(A \dot\cup B) = \alpha + \beta \quad , \quad \#(A \times B) = \alpha \times B \quad \text{and} \quad \#(B^A) = \beta^\alpha \ .$$

E.4 Example

The sets ω, $\omega \dot\cup \omega$ and $\omega \times \omega$ are all the same size (all countably infinite). Well, since we are discussing cardinals here, we should say that the sets $\aleph_0$, $\aleph_0 \dot\cup \aleph_0$ and $\aleph_0 \times \aleph_0$ are all the same size. Therefore, in cardinal arithmetic

$$\aleph_0 = \aleph_0 + \aleph_0 = \aleph_0\aleph_0 \ .$$

(We know that these sets are all different in ordinal arithmetic.)

E.5 Example: More on ℵ notation

The first few infinite cardinals are written $\aleph_0, \aleph_1, \aleph_2, \aleph_3 \ldots$. This extends to $\aleph_\alpha$, for any ordinal α, defined in the obvious way:

$$\begin{aligned} \aleph_0 &= \omega \\ \aleph_{\alpha^+} &= \text{the first cardinal } > \aleph_\alpha \\ \text{For any nonzero limit ordinal } \lambda \qquad \aleph_\lambda &= \sup\{\, \aleph_\xi \,:\, \xi < \lambda \,\} \end{aligned}$$

It is not difficult to prove that this defines a cardinal for every ordinal α and that these are all the cardinals there are.

For any set A of cardinality α, its power set $\mathcal{P}(A)$ is of cardinality 2^α. Thus the Continuum Hypothesis can be restated:

$$\aleph_1 = 2^{\aleph_0}$$

and the Generalised Continuum Hypothesis can be restated: for every infinite cardinal α,

$$\aleph_{\alpha+1} = 2^{\aleph_\alpha} \ .$$

E.6 Theorem: Algebraic properties of cardinal arithmetic

Let α, β and γ be any cardinal numbers. Then the following hold:

(i) $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$

(ii) $\alpha + \beta = \beta + \alpha$

(iii) $\alpha + 0 = \alpha$

(iv) $\alpha(\beta\gamma) = (\alpha\beta)\gamma$

(v) $\alpha\beta = \beta\alpha$

(vi) $0\alpha = 0, \quad 1\alpha = \alpha, \quad 2\alpha = \alpha + \alpha \quad$ and so on.

(vii) $\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$

(viii) $\alpha^{\beta+\gamma} = \alpha^\beta \alpha^\gamma$

(ix) $(\alpha\beta)^\gamma = \alpha^\gamma \beta^\gamma$

(x) $(\alpha^\beta)^\gamma = \alpha^{(\beta\gamma)}$

(xi) $1^\alpha = 1$

(xii) $0^\alpha = 0 \quad$ (provided $\alpha \neq 0$), $\quad 0^0 = 1.$

(xiii) $\alpha^0 = 1, \quad \alpha^1 = \alpha, \quad \alpha^2 = \alpha\alpha, \quad$ and so on.

Proofs. These are all straightforward with the possible exception of (x). Here is a hint. This result boils down to showing that, for any sets A, B and C,

$$\mathcal{F}(C \to \mathcal{F}(B \to A)) \approx \mathcal{F}(B \times C \to A) \ .$$

There is a standard bijection between these two sets. A member of $\mathcal{F}(C \to \mathcal{F}(B \to A))$ is a function $f : C \to \mathcal{F}(B \to A)$. For each such function, let us define a corresponding function $\overline{f} : B \times C \to A$. Given any $b \in B$ and $c \in C$, we define $\overline{f}(b, c) \in A$ as follows: $\overline{f}(b, c) = f(c)(b)$. To see that this makes sense, note that $f(c)$ is a member of $\mathcal{F}(B \to A)$, and so is itself a function from B to A. Thus $f(c)(b)$ makes sense, and is a member of A, as required. It is now necessary to show that $f \mapsto \overline{f}$ does indeed define a bijection. ■

E.7 Theorem: Order properties of cardinal arithmetic

Let α, α' and β be cardinal numbers such that $\alpha \leq \alpha'$. Then the following hold:

(i) $\alpha + \beta \leq \alpha' + \beta$

(ii) $\alpha\beta \leq \alpha'\beta$

(iii) $\alpha^\beta \leq (\alpha')^\beta$

(iv) $\beta^\alpha \leq \beta^{\alpha'}$.

Proofs. These are again straightforward. ■

E.8 Theorem: Finite cardinal arithmetic

For finite cardinals (natural numbers), cardinal, ordinal and ordinary arithmetic are the same.

So far so good. Now we look at a theorem whose corollary tells us that, where infinite cardinals are involved, cardinal arithmetic is mostly trivial.

E.9 Theorem AC

Let α be an infinite cardinal. Then $\alpha^2 = \alpha$ and $2\alpha = \alpha$.

Proof. First let us observe that, if α is an infinite cardinal for which $\alpha^2 = \alpha$, then $\alpha = 2\alpha = 3\alpha$ also. This is easy: since $1 \leq 2 \leq 3 \leq \alpha$, we have $\alpha = 1\alpha \leq 2\alpha \leq 3\alpha \leq \alpha^2 = \alpha$.

Now for the main result. Because of the observation above, it is sufficient to prove that $\alpha = \alpha^2$. In other words, we are trying to prove that $\alpha \approx \alpha \times \alpha$. We consider the subsets of α which have this property, and do a Zorn's Lemma argument on them. Let $\mathcal{A}$ be the set

$$\mathcal{A} = \{\ (A, f)\ :\ A \subseteq \alpha \quad \text{and} \quad f : A \approx A \times A\ \}\ .$$

Since α is infinite, it has a countably infinite subset, N say, and then there is a function $e : N \approx N \times N$. Therefore $\mathcal{A}$ is nonempty and in fact has a member (N, e) with N countably infinite. Our aim is to show that there is some (M, f) in $\mathcal{A}$ such that $M \approx \alpha$; this proves the theorem, for then $\alpha^2 \approx M^2 \approx M \approx \alpha$.

Partially order $\mathcal{A}$ by

$$(A, f) \leq (B, g) \qquad \text{if and only if} \qquad A \subseteq B \quad \text{and} \quad f = g|_A\ .$$

Now let $\mathcal{C}$ be a chain in $\mathcal{A}$. We construct its upper bound in the obvious way:

$$D = \bigcup\{\ C\ :\ (C,f) \in \mathcal{C}\ \} \qquad \text{and} \qquad g = \bigcup\{\ f\ :\ (C,f) \in \mathcal{C}\ \}\ .$$

It is now straightforward to prove that g is a function $D \approx D \times D$ and that, for every $(C,f) \in \mathcal{C}$, $(C,f) \leq (D,g)$. Thus (D,g) is an upper bound for the chain. Applying Zorn's Lemma, $\mathcal{A}$ has a maximal element; call it (M,h).

It remains to show that $M \approx \alpha$. Suppose not. Then $M \prec \alpha$ (since $M \subseteq \alpha$). Now $M \times M \approx M$ and our preliminary observation tells us that $2M \approx M$.

If $\alpha \setminus M \preccurlyeq M$ then $\alpha \approx M + (\alpha \setminus M) \preccurlyeq M + M \approx M$, contradicting $M \prec \alpha$. Therefore $M \prec \alpha \setminus M$, so there is some $E \subseteq \alpha \setminus M$ such that $M \approx E$. By the preliminary observation again, $3E \approx E \approx E^2$.

We now contradict the maximality of (M,h) by finding a function $g\ :\ M \cup E \approx (M \cup E) \times (M \cup E)$ which extends h, so that $(M,h) < (M \cup E, g)$. But this is now easy. $M \cup E$ is a disjoint union, so

$$(M \cup E) \times (M \cup E) \quad \text{is a disjoint sum} \quad (M \times M) \cup (M \times E) \cup (E \times M) \cup (E \times E)\ .$$

Now $M \approx E$, so $M \times E \approx E \times M \approx E \times E \approx E$, and so $(M \times E) \cup (E \times M) \cup (E \times E) \approx 3E \approx E$. Let h' be the bijection $E \to (M \times E) \cup (E \times M) \cup (E \times E)$. Then $h \cup h'$ is a bijection $M \cup E \to (M \times M) \cup (M \times E) \cup (E \times M) \cup (E \times E)$. ■

E.10 Corollary AC

Let α and β be cardinal numbers, $\alpha \leq \beta$ and β be infinite. Then

(i) $\alpha + \beta = \beta$.

(ii) Provided $\alpha \geq 1$ also, $\alpha\beta = \beta$.

(iii) Provided $\alpha \geq 2$ also, $\alpha^\beta = \beta^\beta$.

Proof. (i) $0 \leq \alpha$, so $\beta = 0 + \beta \leq \alpha + \beta \leq \beta + \beta = \beta$.

(ii) $1 \leq \alpha$ so $\beta = 1\beta \leq \alpha\beta \leq \beta^2 = \beta$.

(iii) $2 \leq \alpha \leq \beta$ so $2^\beta \leq \alpha^\beta \leq \beta^\beta \leq (2^\beta)^\beta = 2^{(\beta\beta)} = 2^\beta$. ■

7. MODELS

A Structures, interpretations, models

A language seldom exists in isolation. It is usually used to describe something, perhaps a whole class of similar things. If the language has function symbols $\mathcal{F}$ and relation symbols $\mathcal{R}$, then the thing the language describes presumably has (known) functions and relations to which these symbols refer. This means that this thing must be some kind of set (or class) upon which these functions and relations are defined: we will call this a *structure*. Once we know this, we can ascertain which members of the structure all the terms of the language refer to and thus determine which of the statements in the language are true or false, as they refer to that structure.

In this chapter we will consider only first order theories and the languages which underlie them (first order languages).

Some of the results of interest here will involve languages which are (possibly) uncountably infinite. In this chapter we will allow languages to have arbitrarily large sets of variable, function and relation symbols. As usual, the number of variable symbols will be at least countably infinite.

We recall from Chapter **4** that a first order language $\mathbf{L}$ has ▶ Ch.4

(i) A *function domain*, that is, a set $\mathcal{F}$ of function symbols, each with a prescribed arity,

(ii) a *relation domain*, that is, a set $\mathcal{R}$ of relation symbols, also each with a prescribed arity,

(iii) a set $\mathbf{vL}$ of variable symbols.

Indeed, these three things define the language. Once we have such a language its set of terms is defined as in Chapter 4; here I will call this set $\mathbf{tL}$.

A first-order theory consists of such a language together with a set of axioms, which, with the deductive rules of first order logic, including the notions of proof and deduction, give us our set of theorems.

Then an $(\mathcal{F},\mathcal{R})$-*structure* is a set M upon which there are (actual) functions and relations defined corresponding to the members of $\mathcal{F}$ and $\mathcal{R}$. More precisely,

A.1 Definition: Structure

An $(\mathcal{F},\mathcal{R})$-*structure* M consists of

(i) A nonempty class $|M|$, the *underlying class* or *universe* of M. (In many, but not all, of the structures we consider, $|M|$ is in fact a set, in which case of course we call it the

underlying set.

(ii) To each function symbol f of arity n in $\mathcal{F}$, a function $f_M : |M|^n \to |M|$ (that is, an n-ary operation defined upon $|M|$),

(iii) To each relation symbol r of arity n in $\mathcal{R}$, a relation $r_M \subseteq |M|^n$ (that is, an n-ary relation defined upon M).

Let **L** be a first-order language. Then a *structure for* the language **L** is just an $(\mathcal{F}, \mathcal{R})$-*structure*, where $\mathcal{F}$ and $\mathcal{R}$ are the function and relation domains of the language **L**.

It is important to notice that a structure consists not just of a set or class $|M|$ and the functions and relations on it, but that it also contains a specification of which function corresponds to which function symbol and which relation to which relation symbol. In other words, the functions $f \mapsto f_M$ and $r \mapsto r_M$ are part of the structure (we don't bother to give these functions names because we won't need them). In other words again, if you take a structure M, keep its underlying class $|M|$ but change which actual functions and relations on $|M|$ some of the symbols of the language refer to, you have changed the structure. For this reason I am making a distinction here between M, the class with structure, and $|M|$, the underlying plain old class. It is necessary to make this distinction in case we should ever want to define several alternative structures on the same underlying class. Since we do not often need to do this, we usually ignore this distinction and say, for instance, "m is a member of M" or "g is a function defined on M".

A.2 Examples

(i) The trivial language, $\mathcal{F} = \mathcal{R} = \varnothing$. Any nonempty class whatsoever can act as a structure.

(ii) The *Language of Pure Identity*, in which $\mathcal{F} = \varnothing$ and there is a single binary relation symbol representing equality. Any nonempty class, with the equality symbol representing actual equality, can act as a structure.

▶ 4.B

(iii) The language of **ETA** (Elementary Theory of Arithmetic, Section **4**.B). Here there is a nullary function symbol $\bar{0}$, a unary one, successor, and two binary ones, addition and multiplication. There is a single relation symbol, equality. An obvious structure for this language is $\mathbb{N}$. Alternative structures for this language are the $\mathbb{Z}_n$ (integers mod n) for various n.

▶ 4.C

(iv) The language of the elementary theory of groups (Section **4**.C). Here any (fixed) actual group forms a structure, with the functions and relations of the language interpreted in the obvious way. For a specific example, the symmetric group S_4 is one of the many possible structures for this language.

(v) For a more bizarre example, we could make a structure for the language **VBG** as follows: observing that the language has no function symbols and two binary relation symbols ($\in$ and $=$), we construct a structure by taking the natural numbers $\mathbb{N}$ as its universe, making the relation symbols $\in$ and $=$ correspond to the relations $<$ and ordinary $=$ on $\mathbb{N}$.

A.3 Language versus metalanguage again

In this chapter it will become extremely important to distinguish between expressions in the language being discussed and statements made about the language (statements in the metalanguage).

So I have returned to the convention of using a coloured font for expressions in the object language.

A.4 Remarks leading to the definition of an interpretation

We are about to carefully define how a language can "talk about" one of its structures. In these remarks I want to discuss the idea informally, so the definitions make better sense when they come. It will be convenient to use a slightly bizarre example: let us consider the language **ETA**, the Elementary Theory of Arithmetic, and use as a structure $\mathbb{Z}_5$, the ring of integers modulo 5.

It will be convenient to write $\mathbb{Z}_5$ as the set $\{\,[0]\,,\,[1]\,,\,[2]\,,\,[3]\,,\,[4]\,,\,[5]\,\}$ and use the notation

$$[0] \quad \text{csuc} \quad + \quad \times \quad =$$

for the various functions and relations on it, defined in the usual way. (Here csuc is the "cyclic successor" function $[0] \mapsto [1] \mapsto [2] \mapsto [3] \mapsto [4] \mapsto [5] \mapsto [0]$).

We now have a language, which may be quite limited in what it can express, as is the case with **ETA**. We also have the structure, which again may be quite simple, as is the case here. And then there are we mathematicians, standing outside both of them, and able to describe what is going on with the full power of the normal language of mathematics. We can take a statement in the language, replace its symbols by the corresponding ones in the structure, and so obtain a statement in ordinary mathematical notation; we can then ask, for instance, if this statement is true or false.

There are however some symbols in the language which we have glossed over so far: the logical ones. The symbols $\neg$, $\wedge$, $\vee$, $\Rightarrow$ and $\Leftrightarrow$ of the language all translate to the corresponding logical connectives. Bearing in mind that the idea of a structure for the language is that its underlying class is the "whole universe" as far as the language is concerned, in other words, the language can only "see" its structure and nothing else, we must translate the quantifiers $(\forall x)$ as $(\forall x \in M)$ and $(\exists x)$ as $(\exists x \in M)$, where M is the structure — in our current example $\mathbb{Z}_5$.

As an illustration, let us translate a few of the axioms of the Elementary Theory of Arithmetic as given in Section **4**.B. Axiom NT1, fully quantified for convenience and rewritten in the notation just described, becomes the following sentence in the language ▶ 4.B

$$(\forall x)\neg(\text{suc}(x) = 0)$$

Translating this symbol by symbol as just described, we get

$$(\forall x \in \mathbb{Z}_5)\neg(\text{csuc}(x) = [0])\ .$$

This is a statement in ordinary mathematical language about $\mathbb{Z}_5$; in plainer language, it says that there is no member of $\mathbb{Z}_5$ whose cyclic successor is [0]. This is clearly false in this particular structure, so NT1 says something false about it.

Axiom NT2, rewritten properly, becomes

$$(\forall x)(\forall y)(\,\mathrm{suc}(x) = \mathrm{suc}(y) \;\Rightarrow\; x = y\,)$$

and this translates to

$$(\forall x \in \mathbb{Z}_5)(\forall y \in \mathbb{Z}_5))(\,\mathrm{csuc}(x) = \mathrm{csuc}(y) \;\Rightarrow\; x = y\,)$$

and this is plainly true.

This translation process, when defined properly starting with the next definition, is called *interpretation*.

The two examples above illustrate an obvious but important distinction we must be aware of with such interpretations: as long as the structure has the requisite functions and relations (that is, that it is indeed a structure for the language) then the expressions in the language have interpretations as statements about the structure. However theorems of the theory may or may not translate into true statements about the structure.

There is another important aspect of interpretations which I have not discussed so far: what about expressions with free variables? For example, consider first the expressions without free variables

$$(\exists x)(\,\mathrm{suc}(x) = 0\,) \qquad \text{and} \qquad (\forall x)(\,\mathrm{suc}(x) = 0\,)\,.$$

There is no trouble with either of these: with $\mathbb{Z}_5$ as our structure, the first is true and the second false. But what about this perfectly good expression?

$$\mathrm{suc}(x) = 0 \tag{1A}$$

A natural response is that the truth or falsity (in $\mathbb{Z}_5$) of this expression depends upon which element of that structure x stands for. So let us choose a member of $\mathbb{Z}_5$ for x to stand for and call it $\hat{x}$. Then, with this choice, the interpretation of the expression above is

$$\mathrm{csuc}(\hat{x}) = [0]\;. \tag{1B}$$

If we happen to have chosen $\hat{x}$ to be [4], then this is a true statement about our structure. Any of the other four choices and it would be false.

Another example. Consider the expression

$$(\exists y)(x + y = z)\;. \tag{2A}$$

This one has two free variables and one bound one. To interpret it, we would choose particular values for x and z in our structure — let's call them $\hat{x}$ and $\hat{z}$ — and then interpret the expression as

$$(\exists y \in \mathbb{Z}_5)(\hat{x} + y = \hat{z})\;. \tag{2B}$$

(We see that this will turn out to be true in this structure regardless of the particular choices we make for $\hat{x}$ and $\hat{z}$.) Suppose now we add another quantifier to our expression,

$$(\forall x)(\exists y)(x + y = z)\;. \tag{3A}$$

To interpret this we need only choose a value, $\hat{z}$ say, for z in our structure and then the interpretation would be

$$(\forall x \in \mathbb{Z}_5)(\exists y \in \mathbb{Z}_5)(x + y = \hat{z}) \ . \tag{3B}$$

Note what has happened here: in the passage from (2A) to (3A), we have added one more quantifier $(\forall x)$. In the corresponding passage from (2B) to (3B), we have forgotten our chosen value $\hat{x}$ for x, and replaced it by an ordinary variable which is quantified. Perhaps the best way to think about this is that the role of the symbol x has changed — from a free variable to a bound one — and so the way we interpret it also changes, from a symbol which needs an assigned value before we can interpret the expression to one that doesn't.

While it is clear how this goes, it is necessary to have a proper definition of this process, so that further definitions may be made and facts proved about it. It is natural to define an interpretation by induction over the construction of expressions. This turns out to be a bit tricky, and the definition we will provide shortly may seem a little back-handed. However it is the most straightforward one available.

The easiest way to proceed will be, when defining an interpretation, to start by assuming that values have been chosen in the structure for *all* the variables in the language. Then, when interpreting expressions with quantifiers like (2A) and (3A) above, we simply ignore the chosen values of the quantified variables along with any which do not occur in our expressions.

The upshot of all this is that, for any given language $\mathbf{L}$ and structure M for $\mathbf{L}$ (and except in utterly trivial cases), there are lots of interpretations $\mathbf{L} \to M$, corresponding to different assignments of the variables. Under any particular interpretation, every expression is either true or false, even ones with free variables. However, expressions with free variables will have truth-values which depend on the choice of assignment of the variables and thus on the interpretation, whereas it is obvious that sentences (closed expressions) will have truth-values independent of the particular assignment/interpretation chosen.

In our examples above, the symbols $+$ and suc in our language were made to correspond to functions for which we would naturally use the names $+$ and suc. Of course we are at liberty to make more bizarre choices if it suits our purposes. In Example A.2(v) for instance we interpreted the symbol $\in$ as the relation $<$. Let us look at what happens to a couple of the **VBG** axioms under this interpretation. Consider first the Axiom of Extension, perhaps most conveniently in its fully quantified form

▶ A.2

$$(\forall a)(\forall b)\left((\forall x)(x \in a \Leftrightarrow x \in b) \ \Rightarrow \ a = b\right)$$

The interpretation of this is

$$(\forall a \in \mathbb{N})(\forall b \in \mathbb{N})\left((\forall x \in \mathbb{N})(x < a \Leftrightarrow x < b) \ \Rightarrow \ a = b\right) \ .$$

This is in fact a true statement about $\mathbb{N}$. Now let us look at the Axiom of Power Sets, again in fully quantified form.

$$(\forall a)\left(\mathrm{SET}(a) \ \Rightarrow \ (\exists w)(\,\mathrm{SET}(w) \ \wedge \ (\forall x)(x \subseteq a \Rightarrow x \in w)\,)\right) \ .$$

In order to interpret this we need to replace the symbols SET and $\subseteq$ in this expression by their expansions according to their definitions or, better, to decide what their interpretations

are individually. We have the definitions

$$\begin{array}{rcl} \mathrm{SET}(x) & \Ominus & (\exists s)(x \in s) \\ a \subseteq b & \Ominus & (\forall x)(x \in a \Rightarrow x \in b) \end{array}$$

so these are interpreted as

$$\begin{array}{rcl} \mathrm{SET}_M(x) & \Leftrightarrow & (\exists s \in \mathbb{N})(x < s) \\ a \subseteq_M b & \Leftrightarrow & (\forall x \in \mathbb{N})(x < a \Rightarrow x < b)\ . \end{array}$$

It is not hard to see that $\mathrm{SET}_M(x)$ is always true (for any choice of x), so it is convenient to replace it with the symbol T meaning an always-true expression, and $a \subseteq_M b$ is equivalent to $a \leq b$.

Now we can see that the interpretation of the Power Set Axiom is

$$(\forall a \in \mathbb{N})\left(\mathsf{T} \ \Rightarrow\ (\exists w \in \mathbb{N})(\ \mathsf{T} \ \wedge\ (\forall x \in \mathbb{N})(x \leq a \Rightarrow x < w)\right)\ .$$

This is equivalent to

$$(\forall a \in \mathbb{N})(\exists w \in \mathbb{N})(\forall x \in \mathbb{N})(x \leq a \Rightarrow x < w)\ .$$

Once again, it is easy to see that this is a true statement about our structure $\mathbb{N}$ (let w be any number greater than a). We could proceed a little further and decide how to interpret the power set itself. Its definition (fully quantified as usual) is

$$(\forall a)\big(\mathcal{P}(a)\ =\ \{\, x\ :\ x \subseteq a\,\}\big)$$

which is the same as

$$(\forall x)(\forall a)\big(\, x \in \mathcal{P}(a)\ \Leftrightarrow\ x \subseteq a\,\big)$$

and this is interpreted as

$$(\forall x \in \mathbb{N})(\forall a \in \mathbb{N})\big(\, x < \mathcal{P}_M(a)\ \Leftrightarrow\ x \leq a\,\big)$$

so we can see that, in this structure,

$$\mathcal{P}_M(a) \qquad \text{is} \qquad a + 1\ .$$

The idea of an *interpretation* seems straightforward enough, but is a bit tricky to define. Given our language L and structure M, we start with an assignment of values in M to every variable symbol in L, in other words, a function $\mathsf{vL} \to |M|$. This automatically defines the interpretation of every term in the language as referring to some element of $|M|$, in other words, a function $\mathsf{tL} \to |M|$. Now what is the interpretation of expressions to be? In our informal discussion above, we have translated expressions in our language L into expressions in ordinary mathematical language about the structure. Regarded this way, we have something like a function from the set of expressions in L to the set of expressions in **VBG** (or **ZF** if you prefer). Given any expression in L and interpretation θ, the resulting statement about the structure will be called the *interpretation* of that expression under θ.

One could ask what the status of this "function" is. It cannot be a function in our ordinary mathematical language we are using to discuss the structure, because its values are

statements of that language. If it is to be a function, it must then be a function in the meta-language. It is perhaps better to think of it simply as a well defined rule to create statements about the structure to be discussed.

There is an alternative approach which avoids this excursion into meta-language: we define the "truth-value" of expressions in $\mathbf{L}$ under an interpretation. We will use the set $\{0,1\}$ to stand for truth-values, 0 for false and 1 for true. Let us give it a name:

$$\boldsymbol{\mathcal{T}} \;=\; \{0,1\}\ .$$

As we have seen in the examples above, given an interpretation θ from $\mathbf{tL} \to |M|$, each expression in $\mathbf{L}$ has a corresponding interpretation, which is a statement about the structure which is true or false. We define the truth-value of the expression under θ in such a way that it will be 1 if the interpretation of the expression is a true statement and 0 if it is false. The truth-value of an expression then is a genuine function from the language to $\boldsymbol{\mathcal{T}}$.

These two approaches are entirely equivalent. The first approach (interpreting expressions as statements about the structure) is usually the easier to use whereas the second is more precise and can be used in particularly intricate situations.

(The second approach has another advantage: that it generalises easily to models which use multi-valued logic. For instance, the proof of the independence of the Axiom of Choice uses this method with $\boldsymbol{\mathcal{T}}$ replaced by a boolean algebra.)

So now let us make all this precise.

A.5 Definition: Assignment, interpretation

Let M be a structure for the language $\mathbf{L}$.

(i) An *assignment* is simply a function $\alpha : \mathbf{vL} \to |M|$. (Think of α as assigning a value to each variable symbol in the language.)

We will need the following idea: let $\alpha : \mathbf{vL} \to |M|$ be an assignment, $v \in \mathbf{vL}$ and $m \in M$. Then $\alpha[v/m]$ is the assignment that takes v to m but otherwise agrees with α:

$$\alpha[v/m](x) \;=\; \begin{cases} m & \text{if } x \ominus v, \\ \alpha(x) & \text{otherwise.} \end{cases}$$

(ii) An *interpretation* is a function $\theta : \mathbf{tL} \to |M|$ which has the homomorphism-like property

> for every n-ary function symbol f and terms $t_1, t_2, \ldots, t_n \in \mathbf{tL}$,
> $$\theta(f(t_1, t_2, \ldots, t_n)) \;=\; f_M(\theta(t_1), \theta(t_2), \ldots, \theta(t_n))\ .$$

And note:

A.6 Lemma

With the notation of the previous definition, every assignment $\alpha : \mathbf{vL} \to |M|$ extends uniquely to an interpretation $\bar{\alpha} : \mathbf{tL} \to |M|$.

More carefully: let $\alpha : \mathbf{vL} \to |M|$ be any assignment. Then there is a unique interpretation $\bar{\alpha} : \mathbf{tL} \to |M|$ whose restriction to $\mathbf{vL}$ is α. The values of $\bar{\alpha}$ may be defined inductively over the construction of the terms as follows:

(a) if $v \in \mathbf{vL}$ then $\bar{\alpha}(v) = \alpha(v)$;

(b) if f is an n-ary function symbol and $t_1, t_2, \ldots, t_n$ are terms, then

$$\bar{\alpha}(f(t_1, t_2, \ldots, t_n)) \;=\; f_M(\bar{\alpha}(t_1), \bar{\alpha}(t_2), \ldots, \bar{\alpha}(t_n)) \,.$$

This is a careful definition of what could be said in plainer language thus: given α, we can calculate $\bar{\alpha}(t)$ for any term t by

- replacing every variable symbol v which appears in t by $\alpha(v)$ and
- replacing every function symbol f which appears in t by f_M .

We note also that, if θ is any interpretation $\mathbf{tL} \to |M|$, then $\theta|_{\mathbf{vL}}$, its restriction to $\mathbf{vL}$, is an assignment and clearly θ extends it, so $\theta = \overline{\theta|_{\mathbf{vL}}}$.

These observations allow us to define $\theta[v/m]$ as follows.

A.7 Definition: Substitution in an interpretation

Let M be a structure for the language $\mathbf{L}$ and let $\theta : \mathbf{tL} \to |M|$ be an interpretation. Further, let $v \in \mathbf{vL}$ and $m \in M$. Then $\theta[v/m]$ is the interpretation

$$\theta[v/m] \;=\; \overline{(\theta|_{\mathbf{vL}})[v/m]} \,.$$

The observations above also allow a much easier to understand way of defining such interpretations. Since an interpretation can be defined uniquely by specifying its action on $\mathbf{vL}$, we may define $\theta[v/m]$ by giving its action on $\mathbf{vL}$ only:

$$\theta[v/m](x) \;=\; \begin{cases} m & \text{if } x = v, \\ \theta(x) & \text{otherwise} \end{cases} \qquad \text{for all } v \in \mathbf{vL}.$$

This can also be said in plainer language: given θ, we can calculate $\theta[v/m](t)$ for any term t by

- replacing every occurrence of the variable symbol v which appears in t by m,
- replacing every *other* variable symbol u which appears in t by $\theta(u)$ and
- replacing every function symbol f which appears in t by f_M .

A.8 Definition: Interpretation of an expression

Let $\theta : \mathbf{tL} \to |M|$ be an interpretation and P be an expression in $\mathbf{L}$. Then the *interpretation of P under θ* is a statement defined by induction over P as follows:—

(i) The interpretation under θ of the atomic expression, $r(t_1, t_2, \ldots, t_n)$, where r is an n-ary relation symbol and $t_1, t_2, \ldots, t_n$ are terms, is just the corresponding relation on M:

$$r_M(\theta(t_1), \theta(t_2), \ldots, \theta(t_n)) \,.$$

(ii) The interpretation of $\neg Q$ under θ is $\neg Q$, where Q is the interpretation of Q under θ.

(iii) The interpretation of $Q \Rightarrow R$ under θ is $Q \Rightarrow R$, where Q and R are the interpretations of Q and R under θ respectively.

(iv) The interpretation of $(\forall x)Q(x)$ under θ is $(\forall x \in M)Q(x)$, where $Q(x)$ is the interpretation of $Q(x)$ under $\theta[x/x]$.

A.9 Definition: Truth-value of an expression

(A) Let $\theta : \mathbf{tL} \to |M|$ be an interpretation and P be an expression in $\mathbf{L}$. We define the *truth-value $[\![P]\!]_\theta$ of P under θ* by induction over P:—

(i) if P is an atomic expression, $P \ominus r(t_1, t_2, \ldots, t_n)$ say, where r is an n-ary relation symbol and $t_1, t_2, \ldots, t_n$ are terms, then

$$[\![P]\!]_\theta \;=\; \begin{cases} 1 & \text{if } r_M(\theta(t_1), \theta(t_2), \ldots, \theta(t_n)) \text{ is true} \,, \\ 0 & \text{otherwise.} \end{cases}$$

(ii) if $P \ominus \neg Q$, then

$$[\![]\!]_\theta \;=\; \begin{cases} 1 & \text{if } [\![Q]\!]_\theta = 0 \,, \\ 0 & \text{if } [\![Q]\!]_\theta = 1 \,. \end{cases}$$

(iii) if $P \ominus Q \Rightarrow R$, then

$$[\![P]\!]_\theta \;=\; \begin{cases} 0 & \text{if } [\![P]\!]_\theta = 1 \text{ and } [\![Q]\!]_\theta = 0 \,, \\ 1 & \text{otherwise.} \end{cases}$$

(iv) if $P \ominus (\forall v)Q$, then

$$[\![P]\!]_\theta \;=\; \begin{cases} 1 & \text{if } [\![Q]\!]_{\theta[v/m]} = 1 \text{ for all } m \in M \,, \\ 0 & \text{otherwise.} \end{cases}$$

(B) In this context it is useful to define operators on our set $\mathcal{T}$ of truth-values corre-

sponding tho the various logical operations. For any $a, b, c \in \boldsymbol{\mathcal{T}}$, we define

$$\sim a = \begin{cases} 1 & \text{if } a = 0 \ , \\ 0 & \text{if } a = 1. \end{cases} \qquad \text{corresponding to } \neg \tag{7.1}$$

$$a \multimap b = \begin{cases} 1 & \text{if } a \leq b \ , \\ 0 & \text{if } a = 1 \text{ and } b = 0. \end{cases} \qquad \text{corresponding to } \Rightarrow \tag{7.2}$$

$$a \wedge b = \begin{cases} 1 & \text{if } a = b = 1 \ , \\ 0 & \text{if either } a = 0 \text{ or } b = 0. \end{cases} \qquad \text{corresponding to } \wedge \tag{7.3}$$

$$a \vee b = \begin{cases} 1 & \text{if either } a = 1 \text{ or } b = 1 \ , \\ 0 & \text{if } a = b = 0. \end{cases} \qquad \text{corresponding to } \vee \tag{7.4}$$

$$a \multimap\!\!\circ\!\!\multimap b = \begin{cases} 1 & \text{if } a = b \ , \\ 0 & \text{if } a \neq b. \end{cases} \qquad \text{corresponding to } \Leftrightarrow \tag{7.5}$$

(C) Using this notation we can rewrite Part A of this definition.

If r is an n-ary relation symbol and $t_1, t_2, \ldots, t_n$ are terms, then

$$[\![r(t_1, t_2, \ldots, t_n)]\!]_\theta = \begin{cases} 1 & \text{if } r_M(\theta(t_1), \theta(t_2), \ldots, \theta(t_n)) \text{ is true} \ , \\ 0 & \text{otherwise} \end{cases}$$

and for any expressions P and Q,

$$[\![\neg P]\!]_\theta = \sim [\![P]\!]_\theta \tag{7.6}$$

$$[\![P \Rightarrow Q]\!]_\theta = [\![P]\!]_\theta \multimap [\![Q]\!]_\theta \tag{7.7}$$

$$[\![(\forall v) P]\!]_\theta = \min_{m \in M} [\![Q]\!]_{\theta[v/m]}. \tag{7.8}$$

From this it follows that, for any expressions P and Q,

$$[\![P \wedge Q]\!]_\theta = [\![P]\!]_\theta \wedge [\![Q]\!]_\theta \tag{7.9}$$

$$[\![P \vee Q]\!]_\theta = [\![P]\!]_\theta \vee [\![Q]\!]_\theta \tag{7.10}$$

$$[\![P \Leftrightarrow Q]\!]_\theta = [\![P]\!]_\theta \multimap\!\!\circ\!\!\multimap [\![Q]\!]_\theta. \tag{7.11}$$

We define "P is true in M under θ" (or "P is satisfied by M under θ" or "M satisfies P under θ" etcetera) to mean that $[\![P]\!]_\theta = 1$.

Returning to the remarks preceding this definition, we see that all this can also be described in a plainer way. Given θ and any expression P in L, we can write down an expression for its truth-value $[\![P]\!]_\theta$ under θ in the following way. First, choose an (ordinary mathematical) variable to stand for $\theta(x)$, for each variable symbol x in P (I will use the convention of simply changing font colour, so the variable corresponding to the symbol x is x). Then

- replace every free variable symbol v which appears in P by v (that is, $\theta(v)$) ,
- replace every function symbol f which appears in P by f_M ,

- replace every logical symbol $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$ which appears in P by the corresponding operator $\sim$, $\wedge$, $\vee$, $\rightarrowtriangle$, $\triangleleft\!\!\rightarrowtriangle$ and
- replace every quantifier symbol $(\forall x)$, $(\exists x)$ which appears in P by $\min\limits_{x\in M}$, $\max\limits_{x\in M}$, with the corresponding ordinary variables x.

The result $[\![P]\!]_\theta$ so defined is the *truth-value* of the expression P under θ.

Observe that the only place the interpretation θ figures in this latter version of the definition is in the first dot-point above. Consequently ...

A.10 Lemma

Let P be an expression in the language **L**, θ be an interpretation of **L** in the structure M and P be the interpretation of that expression under θ. Then the statement P is true if and only if the truth-value of P under θ is 1 and false if and only if the truth-value is 0.

Proof. Compare the two definitions above. ■

A.11 Lemma

Suppose that P is an expression in a language **L** and M is a structure for that language.

(i) If θ and φ are two interpretations of **L** in M which agree on all the free variables of P, then the interpretations of P under θ and φ are the same; also the truth-values of P under θ and φ are the same and so P is true in M under θ if and only if it is true in M under φ.

(ii) If P is a sentence (closed expression), then the interpretation of P under all interpretations in M is the same; also the truth-value of P under all interpretations in M is the same and so if P is true in M under any one interpretation then it is true in M under every interpretation. By the same token, if P is false in M under any one interpretation then it is false in M under every interpretation.

A.12 Example: A universe of finite sets

Let us again consider the language **VBG**, but now take as our structure the class of all finite sets, with $\in$ and $=$ interpreted as $\in$ and $=$. Let us call this structure $\mathbb{F}$.

A warning before we start: the members of a finite set need not also be finite. For example, the set $\{\mathbb{Z}, \mathbb{N}\}$ is finite (2 members) whereas its members $\mathbb{Z}$ and $\mathbb{N}$ are not.

What does $\mathrm{SET}(x)$ correspond to? As before, to

$$(\exists s \in \mathbb{F})(x \in s) \ .$$

That is always true: simply set $s = \{x\}$, which is finite.

What about $x \subseteq y$, that is $(\forall u)(\, u \in x \ \Rightarrow\ u \in y)$? This corresponds to

$$(\forall u \in \mathbb{F})(u \in x \ \Rightarrow\ u \in y) \ ,$$

which says that every finite set which is a member of x is also a member of y. This is quite different from $x \subseteq y$. For example, in an interpretation in which x corresponds to$\{\mathbb{Z}, 1, 2\}$ and y to $\{1, 2, 3\}$ (let's call these sets x and y) we see that $x \subseteq y$ is true in $\mathbb{F}$, whereas $x \subseteq y$ is not.

In the light of this, one would expect VBG1 to fail, and so it does. It is interpreted as

$$(\forall a \in \mathbb{F})(\forall b \in \mathbb{F})\big(\,(\forall x \in \mathbb{F})(x \in a \Leftrightarrow x \in b) \;\Rightarrow\; a = b\big).$$

But this is false: for instance take $a = \{\mathbb{Z}\}$ and $b = \varnothing$.

What about VBG2? Given the predicate $P(x)$, it corresponds to

$$(\exists w \in \mathbb{F})(\forall x \in \mathbb{F})(\, x \in w \;\Leftrightarrow\; P(x)\,)$$

where $P(x)$ is the predicate corresponding to $P(x)$. Once more, if we set $P(x)$ to be, say, $x = x$, this is false: it says that there is a finite set w which contains every finite set x.

The failure of VBG1 is a big drawback because it makes the apparatus of definition by description almost unusable — and this is used to define unordered pairs, unions, power sets and, via them, almost everything else. Suppose that nevertheless we are serious about trying to see what the axioms of mathematics have to say in a universe of finite sets; is there any way we can resurrect this example? There are two ways we could go about this:

1 We could redefine $\mathbb{F}$ as the class of all finite sets whose members are also finite, whose members of members are also finite and so on. We simply define a set to be *recursively finite* if it is finite and all its members are recursively finite. This definition looks a bit alarming at first sight, but the Axiom of Foundation tells us that it is valid.

2 We could keep $\mathbb{F}$ as the class of all finite sets, but redefine the interpretation of $=$. There is nothing in the definition of an interpretation to say that **=** must be interpreted as ordinary equality. It is a binary relation symbol and so must be interpreted as a binary relation. The axioms of equality imply that it must be an equivalence relation and "behave well" towards functions and other relations in an obvious way. So, suppose we define an equivalence $\sim$ on $\mathbb{F}$ by:

$$a \sim b \;\Leftrightarrow\; (\forall x \in \mathbb{F})(\, x \in a \;\Leftrightarrow\; x \in b\,)$$

and use this as the interpretation of **=**. Then, having checked that the axioms of equality hold in the new structure, we have more or less forced VBG1 to be true.

It would be difficult, if not impossible, to arrange for VBG2 to be true in such a structure. As an alternative, we could decide to investigate the **ZF** axioms instead. (Note that **ZF** uses the same *language* as **VBG**, it is only the axioms that are different, leading of course to a different theory.) Axiom ZF2 (corresponding to VBG2) states (again for any predicate $P(x)$:

$$(\forall a)(\exists w)(\forall x)(\, x \in w \;\Leftrightarrow\; x \in a \wedge P(x)\,)$$

This translates to

$$(\forall a \in \mathbb{F})(\exists w \in \mathbb{F})(\forall x \in \mathbb{F})(\, x \in w \;\Leftrightarrow\; x \in a \wedge P(x)\,)$$

and this is true in $\mathbb{F}$: given any $a \in \mathbb{F}$, set $w = \{\, x : x \in a \wedge P(x)\,\}$. Since w is a subset of a, it is also finite, and has the required property.

A.13 Lemma

Let $\mathbf{L}$ be a first-order language and M a structure for $\mathbf{L}$. Then $\theta(A)$ is true for every theorem A of $\mathbf{PL}$ and every interpretation $\theta : \mathbf{L} \to M$.

Proof. Axioms PL1–PL3 are

$$\begin{aligned}
&P \;\Rightarrow (Q \Rightarrow P)\\
&(P \Rightarrow (Q \Rightarrow R)) \;\Rightarrow\; ((P \Rightarrow Q) \Rightarrow (P \Rightarrow R))\\
&(\neg P \Rightarrow \neg Q) \;\Rightarrow\; ((\neg P \Rightarrow Q) \Rightarrow P)
\end{aligned}$$

and these translate to

$$\begin{aligned}
&P \;\Rightarrow (Q \Rightarrow P)\\
&(P \Rightarrow (Q \Rightarrow R)) \;\Rightarrow\; ((P \Rightarrow Q) \Rightarrow (P \Rightarrow R))\\
&(\neg P \Rightarrow \neg Q) \;\Rightarrow\; ((\neg P \Rightarrow Q) \Rightarrow P)
\end{aligned}$$

where P, Q and R are the translations of P, Q and R. These are obviously true.

Axioms PL4–PL6 are

$$\begin{aligned}
&(\forall x)(P \Rightarrow Q) \;\Rightarrow\; \big((\forall x)P \Rightarrow (\forall x)Q\big) && \\
&P \;\Rightarrow\; (\forall x)P\,, && \text{where } x \text{ does not occur free in } P\\
&(\forall x)P \;\Rightarrow\; P[x/t]\,, && \text{provided that } [x/t] \text{ is acceptable in } P
\end{aligned}$$

and these translate to

$$\begin{aligned}
&(\forall x \in M)(P \Rightarrow Q) \;\Rightarrow\; \big((\forall x \in M)P \Rightarrow (\forall x \in M)Q\big) && \\
&P \;\Rightarrow\; (\forall x \in M)P\,, && \text{where } x \text{ does not occur free in } P\\
&(\forall x \in M)P \;\Rightarrow\; P[x/t]\,, && \text{provided that } [x/t] \text{ is acceptable in } P
\end{aligned}$$

where here, as before, P, Q and R are the translations of P, Q and R and (in PL6) t is the interpretation of the term t, which must therefore be a member of M. All these are easily proved to be true.

For Modus Ponens, suppose that $\theta(P \Rightarrow Q)$ and $\theta(P)$, that is, $P \Rightarrow Q$ and P, are both true in M. Then so is Q, that is $\theta(Q)$.

Finally, consider Universal Generalisation. It is easier to prove this using the first definition of interpretation of an expression. Suppose that $\theta(P)$ is true in M, for all interpretations θ. To show that $\theta((\forall x)P)$ is true, we must show that $\theta[x/m](P)$ is true in M for all $m \in M$. But this is contained in the assumption. ■

A.14 Definition: Model

Let $\mathbf{L}$ be a language and M a structure for $\mathbf{L}$. Then an expression E of $\mathbf{L}$ is *true for the structure* M if $\theta(E)$ is true for every interpretation $\theta : \mathbf{L} \to M$.

Let L be a language and $\mathcal{E}$ a set of expressions in L. Then a *model* for $\mathcal{E}$ is a structure M such that all the expressions in $\mathcal{E}$ are true for M, that is, true under every interpretation $\theta : \mathsf{L} \to M$. If L is a first order theory, than a *model* for L is a model for its axioms.

If E is a single expression in the language L, we will abuse language slightly and say that M is a model for E if it is a model for $\{E\}$. This is then the same thing as saying that E is true for M.

A.15 Lemma

(i) Every structure is a model for all the theorems of **PL**.

(ii) Let L be a first order theory and $\mathcal{E}$ a set of expressions in L and M be a model for $\mathcal{E}$. If $\mathcal{E} \vdash P$ in L then M is a model for P also. Thus M is a model for all expressions which are deducible from $\mathcal{E}$.

(iii) If L is a first order theory, then M is a model for L if and only if it is a model for its axioms.

(iv) Suppose that L is a first order theory, $\mathcal{E}$ a set of expressions in L and M is a structure for which there exists a interpretation $\theta : \mathsf{L} \to M$ under which not only $\mathcal{E}$ but all expressions deducible from $\mathcal{E}$ are true. Then M is a model for $\mathcal{E}$ (and therefore for all expressions deducible from $\mathcal{E}$).

(v) Suppose that L is a first order theory and M is a structure for which there exists a interpretation $\theta : \mathsf{L} \to M$ under which all theorems of L are true. Then M is a model for L.

(vi) Suppose that L is a first order theory in which all the axioms are sentences (closed expressions) and M is a structure for which there exists a interpretation $\theta : \mathsf{L} \to M$ under which all axioms of L are true. Then M is a model for L.

Proof. This is all quite straightforward.

▶ A.13 **(i)** is a restatement of Lemma A.13.

(ii) is a corollary of (i) and

(iii) is a corollary of (ii).

(iv) By (iii) it is sufficient to show that M is a model for $\mathcal{E}$. Let E be member of $\mathcal{E}$; we want to show that it is true under every interpretation $\theta : \mathsf{L} \to M$. If E contains no free ▶ A.11 variables, then this follows immediately from A.11(ii) above. Suppose now that E contains free variables, $x_1, x_2, \ldots, x_k$ say. Let F be the statement obtained from E by universally quantifying all these variables, i.e. $(\forall x_1)(\forall x_2)\ldots(\forall x_k)E$. Then by Universal Generalisation F is deducible from E and so is true for θ. Also it contains no free variables, so by the argument above is true for every interpretation $\mathsf{L} \to M$, that is, M is a model for F. But $F \Rightarrow E$ by Predicate Logic Axiom PL6. Therefore M is a model for E, by (ii).

(v) is a corollary of (iv).

(vi) See the first part of the proof of (iv). ■

Warnings In parts (ii) and (iii), the passage from a set of expressions to expressions which are deducible from them requires that they be true under *all* interpretations in M, otherwise the result does not follow for deductions involving universal generalisation. In Part (v), the passage from a single interpretation to all interpretations requires that, not only are the axioms true under the given interpretation, but so are all the theorems.

A.16 Definition: Semantic implication

Let P and Q be two expressions in our language $\mathbf{L}$. We will say that P *semantically implies* Q if, for every structure M for $\mathbf{L}$, if P is true for M, then Q is true for M also. This is written $P \models Q$.

More generally, let $\mathcal{P}$ be a set of expressions in our language $\mathbf{L}$. We say that $\mathcal{P}$ *semantically implies* Q if, for every structure M for $\mathbf{L}$, if every expression in $\mathcal{P}$ is true for M, then Q is true for M also. This is written $\mathcal{P} \models Q$.

More generally again, let $\mathcal{P}$ and $\mathcal{Q}$ be sets of expressions in our language $\mathbf{L}$. We say that $\mathcal{P}$ *semantically implies* $\mathcal{Q}$ if, for every structure M for $\mathbf{L}$, if every expression in $\mathcal{P}$ is true for M, then every expression in $\mathcal{Q}$ is true for M also. This is written $\mathcal{P} \models \mathcal{Q}$.

(We note here that the symbol $\models$ is not part of the language $\mathbf{L}$, even informally. It is a metasymbol.)

If Q is a statement in the language $\mathbf{L}$ which is true for every structure for the language, we say that Q is *semantically valid* and we write $\models Q$. This is the same as saying that $\varnothing \models Q$.

(Note that Lemma A.15(i) says that all the theorems of $\mathbf{PL}$ are semantically valid.) ► A.15

A.17 Theorem

For any expressions P and Q in any language $\mathbf{L}$,

$$P \models Q \qquad \text{if and only if} \qquad \models P \Rightarrow Q \ .$$

Proof. Suppose first that $P \models Q$, and let M be any structure for $\mathbf{L}$ and θ any interpretation $\mathbf{L} \to M$. Then it cannot be the case that $\theta(P)$ is true and $\theta(Q)$ is false, because this would contradict $P \models Q$. But that means that $\theta(P) \Rightarrow \theta(Q)$ in M, that is, that $\theta(P \Rightarrow Q)$ is true in M. Since that is true for any M and any θ, this proves that $\models P \Rightarrow Q$.

Conversely, if $\models P \Rightarrow Q$, then $\theta(P) \Rightarrow \theta(Q)$ for every M and θ. But then, if $\theta(P)$ is true, then so is $\theta(Q)$, proving that $P \models Q$. ■

B Adequacy of first order theories

B.1 Theorem

If a first order theory has a model, then it is consistent.

(We grace this statement with the title "Theorem" because of its importance; its proof is easy.)

Proof. Suppose the theory, L say, is inconsistent. That means it has a sentence P (no free variables) such that both $\vdash P$ and $\vdash \neg P$. But then the interpretation of P in the model is both true and false. ■

B.2 Informal definition

► **3**.A.11

We are about to use a new kind of substitution of one symbol for another in expressions. This will be much simpler than the $[v/t]$ kind of substitution defined in **3**.A.11, because now, when substituting t for v, we simply substitute for *every* occurrence of v. In particular, if v happens to be a variable symbol we pay no attention to whether it is bound or free, we substitute for both kinds.

This kind of substitution is much easier to deal with than the $[v/t]$ kind because we do not have to worry about either binding or acceptability. Since it is rather different, let us give it a different name and call it *symbol replacement*.

For example, we could replace x by u in the expression

$$P(x) \;\Rightarrow\; (\forall x)Q(x) \qquad (-1)$$

to get

$$P(u) \;\Rightarrow\; (\forall u)Q(u). \qquad (-2)$$

If u is a variable symbol, the result (–2) is also an expression, but if it is (for instance) a constant symbol the result is no longer an expression (because then $(\forall u)$ is not allowed). Of course, more bizarre replacements are allowed; for example, we could replace $\forall$ by $\neg$ in (–1) to get

$$P(x) \;\Rightarrow\; (\neg x)Q(x) \qquad (-1)$$

which makes no sense at all.

However, the symbol replacements we are going to be concerned with here are only of one simple kind, namely replacing a constant symbol by a variable symbol.

B.3 Lemma

Let L be a first order language in which P is an expression, c a constant symbol and v a variable symbol. Then the result of replacing c by v in P is also an expression.

(In other words, the kind of replacement we are interested in here does not convert expressions into rubbish strings.)

Proof. As exercise. It would be a good idea to convince yourself that you can make a careful proof of this. First define replacement by induction over the construction of terms and expressions (this is almost the same as the "careful" part of Definition **3**.A.11 but much simpler because questions of acceptability do not arise). Then prove this lemma by a similar induction. ■

▶ **3**.A.11

B.4 Lemma

Let **A** be a first order theory, P be a theorem of **A** and let c be a constant symbol which occurs nowhere in the *proper* axioms of **A**. Then there is a variable symbol v such that the result of replacing c by v in P is also a theorem of **A**.

The variable symbol v can be found thus: take any proof $L_1, L_2, \ldots, L_n$ of P and let v be any variable symbol which occurs nowhere in this proof. (Since a proof is of finite length, such a variable symbol must exist.)

Note Here the constant symbol c may appear in some of the axioms P1 – P6 of Predicate Logic; indeed it must do so.

Proof. Assume that the proof $L_1, L_2, \ldots, L_n$ of P and variable symbol v have been chosen as suggested above. Let $L'_1, L'_2, \ldots, L'_n$ be the results of making the replacement $c \to v$ in this given proof. Since now L'_n is the result of replacing c by v in P, it is now enough to show that this new proof is indeed a valid proof. We prove this in the usual way, by showing that each L'_k is either an axiom, or follows from earlier steps by MP or UG.

If L_k is one of the axioms PL1 – PL6 or predicate logic then the result of the replacement in it is to convert it to another instance of the same axiom schema. Perhaps we need to look at this a little more closely. If L_k is an instance of Axiom PL1,

$$L_k \quad \ominus \quad P \;\Rightarrow\; (Q \Rightarrow P)$$

then

$$L'_k \quad \ominus \quad P' \;\Rightarrow\; (Q' \Rightarrow P')$$

(where P' and Q' are defined in the obvious way) and this is obviously another instance of PL1. The proofs for Axioms PL2, PL3 and PL4 are the same. (For PL4 we should observe that x and c are different, since one is a variable and the other a constant.) If L_k is an instance of PL5,

$$L_k \quad \ominus \quad P \;\Rightarrow\; (\forall x)P \qquad \text{where } x \text{ does not ocur free in } P$$

then (again since x and c are different)

$$L'_k \quad \ominus \quad P' \;\Rightarrow\; (\forall x)P'$$

and it is easy to see that x does not occur free in $(\forall x)P'$. If L_k is an instance of PL6,

$$L_k \quad \ominus \quad (\forall x)P \;\Rightarrow\; P[x/t] \qquad \text{where the substitution } [x/t] \text{ is acceptable in } P.$$

Now c is different from t but it might or might not be different from t. In either case we have

$$L'_k \quad \ominus \quad (\forall x)P' \;\Rightarrow\; P'[x/c]$$

and it is not difficult to see that the substitution $[x/c]$ is acceptable in P' also.

That disposes of the axioms PL1 – PL6. If L_k is a proper axiom, then c does not occur in it and so $L'_k \ominus L_k$ and the result is trivial.

Now suppose that L_k follows from two earlier steps by MP. That means that there are two earlier steps of the forms L_i and $L_j \ominus L_i \Rightarrow L_k$. But then the new proof contains earlier steps of the forms L'_i and $L'_j \ominus L'_i \Rightarrow L'_k$ and the result is again trivially true.

Finally, suppose that L_k follows from an earlier step L_i by UG. That means that L_k is of the form $(\forall y)L_i[x/y]$ where the substitution $[x, y]$ is acceptable in L_i. So L'_k is $(\forall y)L'_i[x/y]$ and the replacement $c \to v$ does not stop $[x, y]$ being acceptable. In other words, L'_k follows from L'_i bu UG. ■

B.5 Lemma

Let **A** be a consistent first order theory. Create a new theory **B** by adding a set of new constant symbols, C say, and extending its expressions and the Axioms PL1 – PL6 as necessary to embrace the new constants. Then **B** is a consistent first order theory also.

Note Adding new constants to the set of symbols will result in enlarging the set of expressions. In a first order theory, the **PL** axioms PL1 – PL6 must be schemata, and so now we have more of them too. On the other hand, there is no necessity to enlarge the set of *proper* axioms, and we are not going to do that here.

Proof. **B** is consistent, for if not then there is an expression P of **B** such that $\vdash P \wedge \neg P$ in **B**. The proof of this is finite and so contains only a finite number of variable and constant symbols of **B**. We may therefore replace each constant c which appears in this proof by a variable symbol which did not appear in the proof. By the previous lemma, this transforms the proof into a new correct proof of a statement of the form $P' \wedge \neg P'$ in **A**, contradicting the consistency of **A**. ■

B.6 Theorem

Every consistent first order theory in a countable language has a countable model.

Remarks This theorem is in fact true for languages of any cardinality. I prove the countable case here first since the proof is complicated enough without adding the extra tricks needed to get an arbitrary-cardinality version to work. In the next theorem following I prove the general theorem by adding these tricks.

▶ B.1

With B.1 above, this tells us that a first order theory in a countable language is consistent if and only if it has a countable model.

Proof. Let **A** be the given consistent theory. Create a new theory **B** by adding a denumerably infinite set of new constant symbols $b_1, b_2, b_3, \ldots$ to **A** and extending its expressions and Axioms PL1 – PL6 as necessary to embrace the new constants. By the last lemma, **B** is consistent.

Now let $P_1, P_2, \ldots$ be an enumeration of all the expressions of $\mathbf{B}$ that contain exactly one free variable, and let $x_1, x_2, \ldots$ be their free variables (that is, for each i, x_i is the unique free variable of P_i). Choose a sequence $c_1, c_2, \ldots$ from the new constants $b_1, b_2, \ldots$ in such a way that, for each i, c_i does not occur in any of $P_1, P_2, \ldots, P_i$ and is different from $c_1, c_2, \ldots, c_{i-1}$.

For each i, let Q_i be the expression

$$Q_i \quad \Leftrightarrow \quad P_i[x_i/c_i] \;\Rightarrow\; (\forall x_i)P_i \;. \tag{–1}$$

For each n, let $\mathbf{B}_n$ be the extension of $\mathbf{B}$ obtained by adding $Q_1, Q_2, \ldots, Q_n$ to the axioms of $\mathbf{B}$. Also, let $\mathbf{B}_\infty$ be the extension of $\mathbf{B}$ obtained by adding all the $Q_1, Q_2, \ldots$ to the axioms of $\mathbf{B}$. It is obvious that

$\mathbf{B}_0 = \mathbf{B}$

Every $\mathbf{B}_n$ is an extension of $\mathbf{B}$.

For every $n \geq 1$, $\mathbf{B}_n$ is the extension of $\mathbf{B}_{n-1}$ formed by adding the single axiom Q_n.

$\mathbf{B}_\infty$ is an extension of every $\mathbf{B}_n$.

We prove that every $\mathbf{B}_n$ is consistent by induction. Firstly, $\mathbf{B}_0 = \mathbf{B}$ is consistent by assumption. We now show that, for any $n \geq 1$, if $\mathbf{B}_{n-1}$ is consistent, then so is $\mathbf{B}_n$; this we do by assuming that $\mathbf{B}_n$ is inconsistent and proving that then $\mathbf{B}_{n-1}$ is inconsistent. Making this assumption, any expression at all is provable in $\mathbf{B}_n$; in particular,

$$\vdash \neg Q_n \qquad \text{in } \mathbf{B}_n.$$

But $\mathbf{B}_n$ is obtained from $\mathbf{B}_{n-1}$ by adding one more axiom, namely Q_n. Therefore

$$Q_n \vdash \neg Q_n \qquad \text{in } \mathbf{B}_{n-1},$$

and so

$$\vdash Q_n \Rightarrow \neg Q_n \qquad \text{in } \mathbf{B}_{n-1},$$

from which it follows that

$$\vdash \neg Q_n \qquad \text{in } \mathbf{B}_{n-1}.$$

Using (–1), the definition of Q_n, we see that

$$\vdash P_n[x_n/c_n] \;\wedge\; \neg(\forall x_n)P_n \qquad \text{in } \mathbf{B}_{n-1}$$

and so both

$$\vdash P_n[x_n/c_n] \qquad \text{in } \mathbf{B}_{n-1} \tag{–2}$$

and

$$\vdash \neg(\forall x_n)P_n \qquad \text{in } \mathbf{B}_{n-1}. \tag{–3}$$

Now the constant c_n is not a symbol of $\mathbf{A}$ and so does not occur in any of its axioms. By its choice, it also occurs nowhere in any of $P_1, P_2, \ldots, P_n$ or $c_1, c_2, \ldots, c_{n-1}$. Therefore it occurs nowhere in any of $Q_1, Q_2, \ldots, Q_{n-1}$. This shows that c_n occurs nowhere in any of the proper axioms of $\mathbf{B}_{n-1}$.

Using (–3) and Lemma B.4 above, there is a variable symbol v such that the result of replacing c_n by v in $P_n[x_n/c_n]$ is a theorem of $\mathbf{B}_{n-1}$. ▶ B.4

Now P_n contains no occurrences of c_n and $P_n[x_n/c_n]$ is the result of replacing the *free* occurrences of x_n in P_n by c_n. Consequently, the result of replacing these occurrences of c_n in turn by v is just the same as replacing all free occurrences of x_n in P_n by v (think about it!) — and that is just $P_n[x_n/v]$. This tells us that the result of replacing c_n by v in $P_n[x_n/c_n]$, which we have just seen is a theorem of $\mathbf{B}_{n-1}$, is just $P_n[c_n/v]$. We have proved that

$$\vdash P_n[c_n/v] \qquad \text{in } \mathbf{B}_{n-1}.$$

We now apply UG and conclude that

$$\vdash (\forall v)P_n[c_n/v] \qquad \text{in } \mathbf{B}_{n-1};$$

▶ **3**.D.2 but (–3) gives (using Theorem **3**.D.2)

$$\vdash \neg(\forall v)P_n[x_n/v] \qquad \text{in } \mathbf{B}_{n-1}$$

showing that $\mathbf{B}_{n-1}$ is inconsistent, as required. This completes the induction, showing that every $\mathbf{B}_n$ is consistent.

It follows that $\mathbf{B}_\infty$ is consistent also.

▶ **3**.H.3 We now invoke Theorem **3**.H.3: let $\mathbf{C}$ be a consistent complete extension of $\mathbf{B}_\infty$.

Let us call a term which contains no variable symbols *closed*. Let M be the set of closed terms in $\mathbf{B}$. We make M into a structure (for $\mathbf{B}$ and hence for all the $\mathbf{B}_i$ and $\mathbf{C}$) as follows:

for any n-ary function symbol f,
$$f_M(s_1, s_2, \ldots, s_n) = f(s_1, s_2, \ldots, s_n)$$

(and note that, in particular, each constant symbol b_i is interpreted as itself, $(b_i)_M = b_i$),

for any n-ary relation symbol r,
$$r_M(s_1, s_2, \ldots, s_n) \text{ is true} \quad \text{if and only if} \quad \vdash r(s_1, s_2, \ldots, s_n) \text{ in } \mathbf{C}.$$

We now show that M is a model for $\mathbf{C}$. This will complete the proof, since then it will be a model for $\mathbf{A}$ and is obviously denumerable. To do this, we show that for any sentence (no free variables) C in $\mathbf{C}$,

$$\vdash C \text{ in } \mathbf{C} \qquad \text{if and only if} \qquad C \text{ is true for } M$$

and this we do by induction over the construction of C. If C is an atomic expression, then this is true by the definition of the action of relations on M above. Now we look at the other possibilities.

Suppose first that C is $\neg A$, for some expression A. Then A is a sentence also, so by the inductive hypothesis, A is true for M if and only if $\vdash A$ in $\mathbf{C}$. If C is true for M, then A is false for M and so is not provable in $\mathbf{C}$. But $\mathbf{C}$ is complete, so $\vdash \neg A$ in $\mathbf{C}$, i.e. $\vdash C$ in $\mathbf{C}$, as required. On the other hand, if C is not true for M, then A is true for M and so $\vdash A$ in $\mathbf{C}$. Since $\mathbf{C}$ is consistent, $\neg A$ is not provable in $\mathbf{C}$, i.e. C is not provable in $\mathbf{C}$.

Suppose next that C is $A \Rightarrow B$. (The proof is much the same as for the last case, but here it is anyway.) Since C is a sentence, so are A and B. Then, by the inductive hypothesis, A is true for M if and only if $\vdash A$ in $\mathbf{C}$ and the same is true for B. If C is false for M, then A is true and B false for M. Therefore A is provable and B not provable. But then $\neg B$ is provable and so $\vdash A \wedge \neg B$ from which $\vdash \neg C$ in $\mathbf{C}$ and so, by consistency, C is not provable. If C is true for M, then either A is false or B is true for M. If A is false for M then it is not provable in $\mathbf{C}$ and so, by completeness, $\vdash \neg A$ in $\mathbf{C}$ from which $\vdash A \Rightarrow B$. If B is true for M, then it is provable in $\mathbf{C}$ and again we have $\vdash A \Rightarrow B$.

Now suppose that C is $(\forall x)A$. Since C is a sentence, no variable symbols other than x can occur in A. If x does not occur in A, then $(\forall x)A \dashv\vdash A$ in $\mathbf{C}$ and C is true for M if and only if A is, so the result follows immediately by induction. We may now suppose that A has exactly one free variable x. Then it is one of the expressions $P_1, P_2, \ldots$ listed at the beginning of the proof: there is some k such that $C \ominus (\forall x_k)P_k$.

Let C be true for M. Then, by the definition of a model, P_k is true for every interpretation in M and so, in particular, $P_k[x_k/c_k]$ is true in M. Then, by the inductive hypothesis, $\vdash P_k[x_k/c_k]$ in $\mathbf{C}$. But $\mathbf{C}$ contains, as an axiom, Q_k , that is, $P_k[x_k/c_k] \Rightarrow (\forall x_k)P_k$ and so $\vdash C$ in $\mathbf{C}$.

Finally, let C be false for M. Then, again by the definition of a model, there is some interpretation in M under which P_k is false. Suppose this interpretation maps x_k to t (a member of M and so a closed term of the language), then $P_k([x_k/t])$ is false in M and so, by inductive hypothesis, is not provable in $\mathbf{C}$. But then $(\forall x_k)P_k$, that is C, is not provable either. ■

This innocuous-looking theorem has a most extraordinary corollary ...

B.7 Example: A countable model of VBG Set Theory!

Assuming that **VBG** is consistent, it has a countable model. The same is true of **ZF**.

This seems paradoxical — Set Theory contains assertions of the existence of uncountable infinite sets, $\mathcal{P}(\mathbb{N})$ for example. How can such assertions be true in a countable model?

A model, M say, of **VBG** contains members corresponding to sets and a binary relation corresponding to the relation symbol $\in$. However this binary relation on M need not itself be set membership — it probably won't be — so let us denote it $\lhd$. Let P be the member of M corresponding to $\mathcal{P}(\mathbb{N})$. Since M is countable the set of all x in M such that $x \lhd P$ is countable. Thus, standing outside the theory, we can construct a one-to-one function between $\mathbb{N}$ and these "members" x of P. However there is nothing *in the model* which does the job of this function.

B.8 The "Downward" Löwenheim-Skolem Theorem AC

Every consistent first order theory in a language $\mathbf{K}$ has a model of cardinality no greater than that of $\mathbf{K}$.

Remark

▶ B.1 This theorem, together with B.1 above, tells us that a first order theory is consistent if and only if it has a model.

Proof. Let $\mathbf{A}$ be the given theory in a language $\mathbf{K}$ of cardinality κ. Create a set $B = \{b_i\}_{i<\kappa}$ of the same cardinality as $\mathbf{K}$ and disjoint from it. Let $\mathbf{L}$ be the new language formed from $\mathbf{K}$ by adding (all the members of) B as new constants to the language and extending the language as necessary. Observe that $\mathbf{L}$ is also of cardinality κ. Let $\mathbf{B}$ be the correspondingly
▶ B.5 extended theory. Now $\mathbf{B}$ is consistent by Lemma B.5.

The set of all expressions in $\mathbf{B}$ which have at most one free variable is easily seen to be of cardinality κ, so let us index it thus: $\{P_i(x_i)\}_{i<\kappa}$. Define a sequence $\{c_i\}_{i<\kappa}$ in B by: for each $i < \kappa$, c_i is the first member of B that does not occur in any of the $\{P_j(x_j)\}$ for $j < i$ and is different from all the $\{c_j\}$ for $j < i$.

(How do we know that such a $\{c_i\}$ exists at all? Let i^* be the cardinality of i and note that $i^* \leq i < \kappa$. The set of all members of B which occur in any of the $\{P_j(x_j)\}$ for $j < i$ is the union of i^* finite sets and so is of cardinality $\leq i^*$. The set of all the $\{c_j\}$ for $j < i$ is of cardinality i^*. So the set of all members of B which may not be chosen, being the union of these two sets, is of cardinality i^*. But B is of cardinality $\kappa > i^*$, so there must be members left over.)

Now, for each $i < \kappa$, let Q_i be the expression

$$Q_i \quad \Leftrightarrow \quad P_i(c_i) \;\Rightarrow\; (\forall x_i) P_i(x_i)$$

Define a sequence of theories $\{\mathbf{B}_\nu\}_{\nu\leq\kappa}$ (all with the same language $\mathbf{L}$) by: $\mathbf{B}_\nu$ is the theory obtained by adding all of $\{Q_i\}_{i<\nu}$ to the axioms of $\mathbf{B}$. It is obvious that

$\mathbf{B}_0 = \mathbf{B}$

Every $\mathbf{B}_\nu$ is an extension of $\mathbf{B}$.

For every ν, $\mathbf{B}_{\nu+1}$ is the extension of $\mathbf{B}_\nu$ obtained by adding the one extra axiom Q_ν.

If ν is a nonzero limit ordinal, then $\mathbf{B}_\nu = \bigcup\{\, \mathbf{B}_i \,:\, i < \nu \,\}$.

For all i and j such that $i < j \leq \kappa$, $\mathbf{B}_j$ is an extension of $\mathbf{B}_i$.

And so $\mathbf{B}_\kappa$ is an extension of every $\mathbf{B}_i$.

Next we check that every $\mathbf{B}_\nu$ is consistent, by induction over ν. We have already proved that $\mathbf{B}_0 = \mathbf{B}$ is. Suppose that $\mathbf{B}_\nu$ is consistent; the proof that then so is $\mathbf{B}_{\nu+1}$ is exactly the same as in the countable version of the proof. If ν is a nonzero limit ordinal and $\mathbf{B}_i$ is consistent for all $i < \nu$, then $\mathbf{B}_\nu$, being the union of these, is also consistent by the usual argument (any proof of $P \wedge \neg P$ in $\mathbf{B}_\nu$ must involve only a finite number of axioms, and they must all belong to some $\mathbf{B}_\mu$ with $\mu < \nu$.)

We now know that $\mathbf{B}_\kappa$ is consistent. Let $\mathbf{C}$ be a consistent complete extension of $\mathbf{B}_\kappa$. The remainder of the proof is exactly the same as for the countable version of the theorem above, noting that M, being a subset of $\mathbf{L}$, is of cardinality no greater than κ. ■

(This proof only requires the Axiom of Choice in its tacit assumption that every language does in fact have a cardinality. If we do not accept the Axiom of Choice, and so do not accept that every set necessarily has a cardinality, then we can interpret this theorem to apply only to languages that do happen to have a cardinality. In that case the Axiom of Choice is not involved.)

B.9 The "Upward" Löwenheim-Skolem Theorem AC

(Also called the Löwenheim-Skolem-Tarski Theorem or the LST Theorem.)

Every consistent first order theory in a language **K** has a model of every cardinality greater than or equal to that of **K**.

Proof. Let us suppose that the cardinality of the language **K** is κ and that λ is some cardinal $\geq \kappa$; we construct a model of cardinality λ.

Repeat the proof of Theorem B.8 above with one change: where we create the set B at the very beginning, now create it $\{b_i\}_{i<\lambda}$ of cardinality λ. The whole of the rest of the proof proceeds unchanged. ▶ B.8

Observe at the end that, since $B \subseteq M \subseteq \mathbf{L}$, it follows that M has cardinality λ. ■

(It is possible to rewrite this theorem in a way which does not require the Axiom of Choice: Every consistent first order theory in a language **K** which has a cardinality has a model of every cardinality greater than or equal to that of **K**.)

This theorem also has some surprising corollaries, for instance:

B.10 Corollary

There is a model of First Order Number Theory which is uncountable. ■

(Note that this one does not require the Axiom of Choice.)

B.11 Definition: Models respecting equality

If **L** is a first order theory with equality and M is a structure for **L**, then M must have a binary relation $=_M$, but in a general structure this need not be ordinary equality.

The axioms of equality force the relation $=_M$ to be an equivalence relation; not only that, but an equivalence relation with special properties. It must, for instance behave well towards the functions of the structure, in the sense that, for any n-ary function symbol f and terms $x_1, x_2, \ldots, x_n, x'_1, x'_2, \ldots, x'_n$,

$$\text{if} \quad x_1 = x'_1, x_2 = x'_2 \ldots, x_n = x'_n \quad \text{then} \quad f_M(x_1, x_2, \ldots, x_n) \;=_M\; f_M(x'_1, x'_2, \ldots, x'_n)\,.$$

It must also behave well in a similar fashion towards all relations and indeed to all expressions.

However none of this guarantees that $=_M$ must actually be equality itself. In many cases it does of course and this is a very useful property of a model and so we give it a name:

A model M *respects equality* if the relation $=_M$ is ordinary equality.

(We could say, if $=_M$ $==$, but that would be silly!)

B.12 Theorem

If a first order theory with equality has a model then it has a model which respects equality.

More specifically, if it has a model of cardinality κ, then it has a model of cardinality no greater than κ which respects equality.

Proof. Let $\mathbf{A}$ be a first order theory in a language $\mathbf{L}$ with equality and M be a model for $\mathbf{A}$. We use M to construct another model M' which respects equality.

Step 1 Define a relation $\sim$ on M by: $a \sim b$ if either $a = b$ or, for every interpretation $\theta : \mathbf{tL} \to M$, there are terms $s, t \in \mathbf{tL}$ such that $\vdash s = t$, $\theta(s) = a$ and $\theta(t) = b$.

Note that it is obvious that $\sim$ is reflexive and symmetric, however it may not be transitive.

Now suppose that $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ are members of M and that $a_i \sim b_i$ for $i = 1, 2, \ldots, n$. Then, for any n-ary function symbol f,

$$f_M(a_1, a_2, \ldots, a_n) \sim f_M(b_1, b_2, \ldots, b_n)$$

and, for any n-ary relation symbol r,

$$r_M(a_1, a_2, \ldots, a_n) \Leftrightarrow r_M(b_1, b_2, \ldots, b_n)$$

Proof of this: Let θ be any interpretation of $\mathbf{A}$ in M. Then, for each $i = 1, 2, \ldots, n$, either $a_i = b_i$ or there are terms s_i, t_i such that,

$$\theta(s_i) = a_i, \quad \theta(t_i) = b_i \quad \text{and} \quad \vdash s_i = t_i \,.$$

Then, by the axioms of equality,

$$f(s_1, s_2, \ldots, s_n) = f(t_1, t_2, \ldots, t_n) \quad \text{and} \quad r(s_1, s_2, \ldots, s_n) \Leftrightarrow r(t_1, t_2, \ldots, t_n) \,.$$

Since $f_M(a_1, a_2, \ldots, a_n) = \theta(f(s_1, s_2, \ldots, s_n))$ and $f_M(b_1, b_2, \ldots, b_n) = \theta(f(t_1, t_2, \ldots, t_n))$ the definition of $\sim$ tells us that

$$f_M(a_1, a_2, \ldots, a_n) \sim f_M(b_1, b_2, \ldots, b_n) \,.$$

Also $r_M(a_1, a_2, \ldots, a_n) \Leftrightarrow r_M(b_1, b_2, \ldots, b_n)$ by the definition of r_M.

Step 2 Now we define another relation $\equiv$ on M by: $a \equiv b$ if there is a finite sequence $u_1, u_2, \ldots, u_k$ $(k \geq 0)$ in M such that $u_0 = a$, $u_k = b$ and $u_{i-1} \sim u_i$ for $i = 1, 2, \ldots, k$. This is the "closure of $\sim$ to an equivalence relation": it is an easy matter to check that this is indeed an equivalence relation. Further, it is a congruence relation, in the sense that it has the following properties inherited from $\sim$: If $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ are members of M such that $a_i \equiv b_i$ for $i = 1, 2, \ldots, n$ then, for any n-ary function symbol f,

$$f_M(a_1, a_2, \ldots, a_n) \equiv f_M(b_1, b_2, \ldots, b_n)$$

and, for any n-ary relation symbol r,

$$r_M(a_1, a_2, \ldots, a_n) \Leftrightarrow r_M(b_1, b_2, \ldots, b_n) \ .$$

Proof of this: From the definition of $\equiv$, there are natural numbers $k_1, k_2, \ldots, k_n$, all ≥ 0, and members $u_{i,j}$ of M for $i = 1, 2, \ldots, n$ and $j = 0, 1, \ldots, k_i$ such that $a_i = u_{i,0}$ and $b_i = u_{i,k_i}$ for $i = 1, 2, \ldots, n$ and $u_{i,j-1} \sim u_{i,j}$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k_i$. The trickery with subscripts here is because the chains of u's connecting a_i to b_i may be of different lengths for different i. However, we can always make any of these chains longer, by simply repeating the last u, using the fact that $\sim$ is reflexive. So we may, in fact, assume that all these chains are the same length, which is the same as assuming that all the k_i are the same. So let us do this; now we have *one* natural number k and members $u_{i,j}$ of M for $i = 1, 2, \ldots, n$ and $j = 0, 1, \ldots, k$ such that $a_i = u_{i,0}$ and $b_i = u_{i,k}$ for $i = 1, 2, \ldots, n$ and $u_{i,j-1} \sim u_{i,j}$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$. But now, from what was proved above about $\sim$,

$$\begin{aligned} & f_M(u_{1,j-1}, u_{2,j-1}, \ldots, u_{n,j-1}) \ \sim \ f_M(u_{1,j}, u_{2,j}, \ldots, u_{n,j}) \\ \text{and} \quad & r_M(u_{1,j-1}, u_{2,j-1}, \ldots, u_{n,j-1}) \ \Leftrightarrow \ r_M(u_{1,j}, u_{2,j}, \ldots, u_{n,j}) \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for } = 1, 2, \ldots, k. \end{aligned}$$

From this

$$f_M(a_1, a_2, \ldots, a_n \ = \ f_M(u_{1,0}, u_{2,0}, \ldots, u_{n,0}) \ \equiv \ f_M(u_{1,k}, u_{2,k}, \ldots, u_{n,k}) \ = \ f_M(b_1, b_2, \ldots, b_n)$$

and

$$r_M(a_1, a_2, \ldots, a_n) \ = \ r_M(u_{1,0}, u_{2,0}, \ldots, u_{n,0}) \ \Leftrightarrow \ r_M(u_{1,k}, u_{2,k}, \ldots, u_{n,k}) \ = \ r_M(b_1, b_2, \ldots, b_n)$$

as required.

Step 3 Now, let M' be the set of all congruence classes in M' (equivalence classes under the congruence $\equiv$). For any member x of M, we will write $[x]$ for the congruence class of x. We also write $\pi : M \to M'$ for the "natural projection" $x \mapsto [x]$.

We make M' into a structure for $\mathbf{L}$ as follows. For any n-ary function symbol f, define $f_{M'} : M' \to M'$ thus: let $X_1, X_2, \ldots, X_n$ be members of M', that is, congruence classes in M. Choose representatives $x_1, x_2, \ldots, x_n$ from these classes. Define $f_{M'}(X_1, X_2, \ldots, X_n)$ to be the congruence class containing $f_M(x_1, x_2, \ldots, x_n)$. The results just proved above tell us that the particular choice of representatives $x_1, x_2, \ldots, x_n$ in their classes is irrelevant. The definition then can be written succinctly:

$$f_{M'}([x_1], [x_2], \ldots, [x_n]) \ = \ [f_M(x_1, x_2, \ldots, x_n)].$$

In the same way we define

$$r_{M'}([x_1], [x_2], \ldots, [x_n]) \ \Leftrightarrow \ r_M(x_1, x_2, \ldots, x_n)$$

It follows immediately from this definition that, if θ is an interpretation $\mathbf{L} \to M$, then $\pi \circ \theta$ is an interpretation $\mathbf{L} \to M'$.

Step 4 We now show that, if P is any expression in $\mathbf{L}$ and θ any interpretation $\mathbf{L} \to M$,

$$P \text{ is true in } M \text{ under } \theta \quad \text{ if and only if } \quad \text{it is true in } M' \text{ under } \pi \circ \theta.$$

We do this by induction over the construction of P using the formal definition in A.9. ► A.9

(i) If $P = r(t_1, t_2, \ldots, t_n)$ then it is true in M under θ if and only if

$$r_M(\theta(t_1), \theta(t_2), \ldots, \theta(t_n)) \quad \text{is true}$$

and it is true in M' under $\pi \circ \theta$ if and only if

$r_{M'}(\pi\circ\theta(t_1), \pi\circ\theta(t_2), \ldots, \pi\circ\theta(t_n))$ is true, that is, $r_{M'}([\theta(t_1)], [\theta(t_2)], \ldots, [\theta(t_n))]$ is true

and we proved these things equivalent in the previous step.

(ii) Suppose that P is $\neg Q$ and (inductively), for every interpretation $\theta : \mathbf{L} \to M$, Q is true in M under θ if and only if it is true in M' under $\pi \circ \theta$. Then, for any such θ,

$$\begin{aligned} P \text{ is true in } M \text{ under } \theta &\Leftrightarrow Q \text{ is not true in } M \text{ under } \theta \\ &\Leftrightarrow Q \text{ is not true in } M' \text{ under } \pi \circ \theta \\ &\Leftrightarrow P \text{ is true in } M' \text{ under } \pi \circ \theta \end{aligned}$$

(iii) In the case in which P is $Q \Rightarrow R$ the argument is the same.

(iv) Suppose that P is $(\forall v)Q$ and, as usual, that, for every interpretation $\theta : \mathbf{L} \to M$, Q is true in M under θ if and only if it is true in M' under $\pi \circ \theta$. Then for any such θ,

$$\begin{aligned} P \text{ is true in } M \text{ under } \theta &\Leftrightarrow Q \text{ is true in } M \text{ under } \theta_{[v/m]} \text{ for every } m \in M \\ &\Leftrightarrow Q \text{ is true in } M' \text{ under } \pi \circ \theta_{[v/m]} \text{ for every } m \in M \\ &\Leftrightarrow Q \text{ is true in } M' \text{ under } (\pi \circ \theta)_{[v/[m]]} \text{ for every } m \in M \\ &\Leftrightarrow Q \text{ is true in } M' \text{ under } (\pi \circ \theta)_{[v/m']} \text{ for every } m' \in M' \\ &\Leftrightarrow P \text{ is true in } M' \text{ under } \pi \circ \theta \end{aligned}$$

In the third equivalence here we use the fact that $\pi \circ \theta_{[v/m]} = (\pi \circ \theta)_{[v/[m]]}$, which can be checked by verifying that both these interpretations have the same restrictions to $\mathbf{vL}$. The fourth equivalence depends upon the fact that π is *onto* M'.

Step 5 Now we show that M' is a model for $\mathbf{A}$. Suppose $\vdash P$ in $\mathbf{A}$ and θ is any interpretation $\mathbf{A} \to M'$. Let us write α for the restriction of θ to the set $\mathbf{vL}$ of variables of $\mathbf{L}$ (this is the assignment which defines θ). Since the natural projection π is onto M', we may "factor α through π", that is, there is a function $\beta : \mathbf{vL} \to M$ such that $\alpha = \pi \circ \beta$. Now β is an assignment $\mathbf{vL} \to M$, and so can be extended uniquely to an interpretation, ψ say, $\mathbf{L} \to M$. Now ψ, restricted to $\mathbf{vL}$ is β, so $\pi \circ \psi$, restricted to $\mathbf{vL}$ is $\pi \circ \beta = \alpha$. Since an interpretation is defined by its restriction to the set of variable symbols, it follows that $\pi \circ \psi = \theta$. Since M is a model, P is true in M under θ. Now it follows from the previous step that it is also true in M' under ψ.

Step 6 The new model M' respects equality: if $\vdash s = t$ and θ is any interpretation $\mathbf{L} \to M'$, construct $\psi : \mathbf{L} \to M$ so that $\theta = \pi \circ \psi$, as in the previous step. Then $\psi(s) \sim \psi(t)$, so $\psi(s) \equiv \psi(t)$, so $[\psi(s)] = [\psi(t)]$, that is $\theta(s) = \theta(t)$.

Also, the function $\pi : M \to M'$ is onto, so the cardinality of M' is no greater than that of M.

(Note that the Axiom of Choice is used in this last statement. It has also been used earlier in the proof — in Step 5 it is required to factor α through π.) ■

B.13 Remark

It follows from this theorem that any consistent theory has a model which respects equality, however the theorem as proved is not so specific about the cardinality of the model.

For example, a countable theory has a countable (ordinary) model M, and then the theorem constructs the model which respects equality as the quotient model M', however this may be much smaller than M.

Here is an easy example. Choose any particular positive integer, say 3. Then it is easy enough to construct a theory with axioms which ensure that a model which respects equality must have exactly three members. For example, include three (different) constant symbols, a, b and c say and the axioms

$$\neg(a = b)$$
$$\neg(a = c)$$
$$\neg(b = c)$$
$$(\forall x)(\,(x = a) \vee (x = b) \vee (x = c)\,)$$

If the model is not required to respect equality, we may construct an infinite model as above. The only effect axioms such as these would have is to force every member of the model to stand in the relation $=_M$ to one of the three elements which interpret a, b and c — but this relation is not actual equality.

B.14 The Adequacy Theorem, often called "Gödel's Completeness Theorem")

Any first order theory is adequate.

In other words, in any first order theory, the theorems are exactly those expressions which are semantically implied by the axioms.

In other words again, if **A** is a first order theory with axioms $\mathcal{A}$ and P is any expression in **A**, then

$$\mathcal{A} \vdash P \qquad \text{if and only if} \qquad \mathcal{A} \models P\ .$$

Proof. That if $\mathcal{A} \vdash P$ then $\mathcal{A} \models P$ is just a restatement of Lemma A.15(ii). ▶ A.15

Now suppose that P is not provable in **A** (that is, that it is not the case that $\mathcal{A} \vdash P$). Form the extension **B** of **A** by adding $\neg P$ as a new axiom to $\mathcal{A}$. We know that **B** is consistent. By Theorem B.8, **B** has a model. This is also a model for **A**, and in it P is not true. ■ ▶ B.8

B.15 The Adequacy Theorem for first order theories with equality

If **A** is a first order theory with equality and axioms $\mathcal{A}$ and P is any expression in **A**, then

$$\mathcal{A} \vdash P \qquad \text{if and only if} \qquad \mathcal{A} \models_{=} P$$

where $\models_{=}$ means true under every structure which respects equality).

Proof. To see that, if $\mathcal{A} \vdash P$ then $\mathcal{A} \models_{=} P$ observe that, if $\mathcal{A} \vdash P$ then, by the previous theorem, $\mathcal{A}$ semantically implies P in every structure, including those which respect equality.

The other part of the argument is the same as for the previous theorem. Suppose that P is not provable in $\mathbf{A}$ (that is, that it is not the case that $\mathcal{A} \vdash P$). Form the extension $\mathbf{B}$ of $\mathbf{A}$ by adding $\neg P$ as a new axiom to $\mathcal{A}$. We know that $\mathbf{B}$ is consistent. By Theorem B.12, $\mathbf{B}$ has a model which respects equality. This is also a model for $\mathbf{A}$, and in it P is not true. ■

▶ B.12

C Compactness

C.1 Theorem

Let **L** be a first order language, $\mathcal{A}$ be a set of statements in **L**. If every finite subset of $\mathcal{A}$ generates a consistent theory then so does $\mathcal{A}$.

Proof. This is straightforward — we have met this idea before. If $\mathcal{A}$ does not generate a consistent theory, then there is a statement P such that $\mathcal{A} \vdash P \wedge \neg P$. There can be only a finite number of members of $\mathcal{A}$ involved in the proof of this, so that finite subset generates an inconsistent theory. ■

C.2 The Compactness Theorem

Let **L** be a first order language, $\mathcal{A}$ be a set of statements in **L**. If every finite subset of $\mathcal{A}$ has a model then so does $\mathcal{A}$.

Proof. This is a corollary of the previous theorem, using B.8. ■ ▶ B.8

C.3 Example: A model of arithmetic with an infinite number

In Section **4**.B we discussed the Elementary Theory of Arithmetic, **ETA**. The structure which immediately springs to mind for this system is the natural numbers, together with the usual zero, successor, addition and multiplication functions, which we could call $(\mathbb{N}, \bar{0}, {}^{+}, +, .)$. ▶ 4.B

Let us now extend this language by adding a constant $\bar{\omega}$ and a countably infinite set of new axioms

$$\bar{0} < \bar{\omega} \quad , \quad \bar{1} < \bar{\omega} \quad , \quad \bar{2} < \bar{\omega} \quad , \ \ldots$$

(where, as usual, $\bar{1}, \bar{2}, \bar{3}, \ldots$ are abbreviations for $\bar{0}^{+}, \bar{0}^{++}, \bar{0}^{+++}, \ldots$ and $a < b$ is an abbreviation for $(\exists x)(a + x^{+} = b)$). We could call this new language $\mathbf{ETA}_{\omega}$.

It is important to observe here that we are *extending* the Elementary Theory of Arithmetic: we make the language by using all the symbols of that theory and adding the new one $\bar{\omega}$ and also using all of its seven axioms (NT1) – (NT7) and adding the new ones above. Because of this we are retaining the axiom of induction amongst other things.

Now if $\mathcal{B}$ is any *finite* subset of these axioms, then it can only contain a finite number of the new axioms listed above. Thus there is a last one, $\bar{n} < \bar{\omega}$ say, which occurs in $\mathcal{B}$. Then we have a model for $\mathcal{B}$: the (ordinary) natural numbers, with the usual interpretation of all the symbols other than $\bar{\omega}$, and interpreting $\bar{\omega}$ as $n + 1$.

It follows that the theory itself has a model. But now the new axioms, taken together, state that ω is greater than every one of the "ordinary" natural numbers. Note that, because of the other axioms, the model must contain a whole host of "infinite" numbers, corresponding to $\bar{\omega}^{+}$, $\bar{2}\bar{\omega}$, $\bar{\omega}\bar{\omega}$ and so on. Note also that, while a model for $\mathbb{N}$ together with an infinite number ω is not surprising, one in which the principle of (ordinary) induction holds for the whole set is!

How can this be? Consider, for instance, the expression $x < \hat{\omega}$. Now $\vdash 0 < \hat{\omega}$ in our theory, and a glance at the new axioms above tells us that $\vdash (\forall x)(x < \hat{\omega} \Rightarrow x^+ < \hat{\omega})$ also. So the principle of induction tells us that $\vdash (\forall x)(x < \hat{\omega})$; and therefore $\vdash \hat{\omega} < \hat{\omega}$, which is absurd (a contradiction). SO, there is something wrong with this argument — what is it?

The answer to this question is that, in the argument above, I jumped too quickly to a conclusion, namely that

$$\vdash \quad (\forall x)(x < \hat{\omega} \Rightarrow x^+ < \hat{\omega})\,. \tag{!!}$$

What we have, from the axioms, is that

$$\text{for all } x \in \mathbb{N} \quad \vdash \quad x < \hat{\omega}$$

and to get from here to the supposed theorem would involve all those infinite number of axioms, and so an infinitely long proof: and there is no such thing. Now one might imagine that there was some sneaky way of getting around this problem, and proving the theorem labelled (!!) some other way. But this cannot be, because if this were a theorem, the rest of the argument above is watertight and we would indeed have $\vdash \hat{\omega} < \hat{\omega}$, which is an antitheorem (by the definition of $<$). And that would show our new theory $\mathbf{ETA}_\omega$ to be inconsistent, contradicting the Compactness Theorem.

C.4 Example: A model of the reals with infinite numbers and infinitesimals

We can apply a similar idea to the real numbers to add infinitesimals. To do this we are going to represent $\mathbb{R}$ by a language having uncountably infinite sets of function and relation symbols and an uncountably infinite number of axioms.

We define our language, $\mathbf{L}$ say, to have a function symbol $\hat{f}$ of arity n corresponding to *every* actual function $f : \mathbb{N}^n \to \mathbb{N}$ — the whole $2^{2^{\aleph_0}}$ of them. Also a relation symbol $\hat{r}$ of arity n corresponding to every actual relation of arity n, that is, every subset of $\mathbb{R}^n$, again $2^{2^{\aleph_0}}$ of them.

For our theory we simply take every sentence (closed expression) in this language which is true in $\mathbb{R}$, and we take the axioms to be the same set of expressions: there's no harm in that, even if it is a bit unusual.

So now we have an enormous first order system. Nevertheless, it is in many ways quite easy to deal with. For example, it is obvious, from the way it was built, that $\mathbb{R}$ (with all its functions and relations) is a model.

Now we add an infinitesimal in much much the same way as we added an infinity to **ETA**. We add a new constant (nullary function) ι and axioms

$$\begin{aligned} \hat{0} &< \iota \\ \iota &< \hat{r} \qquad \text{for every positive real number } r\,. \end{aligned}$$

As with the previous example, $\mathbb{R}$ is a model for every finite set of these axioms. Consequently the extended language has a model.

This model is the "nonstandard reals" and is used for Nonstandard Analysis.

C.5 Theorem

If a first order theory has arbitrarily large finite models which respect equality, then it has an infinite model which respects equality.

Preliminary Suppose that $P(z_1, z_2, \ldots, z_n)$ is an expression involving free variables $z_1, z_2, \ldots, z_n$, that

$$(\exists z_1)(\exists z_2)\ldots(\exists z_n)\, P(z_1, z_2, \ldots, z_n)$$

has occurred as a line in a proof and that $y_1, y_2, \ldots, y_n$ are other variable symbols which have not occurred anywhere in the proof so far (including its hypotheses, if any). Then we may apply the choice rule first to z_1 to obtain

$$(\exists z_2)(\exists z_3)\ldots(\exists z_n)\, P(y_1, z_2, z_3, \ldots, z_n)\ ,$$

and then to z_2 to obtain

$$(\exists z_3)\ldots(\exists z_n)\, P(y_1, y_2, z_3, \ldots, z_n)\ ,$$

and so on, after n such steps obtaining

$$P(y_1, y_2, \ldots, y_n)\,.$$

In the proof below we will apply this to the expression

$$(\exists z_1)(\exists z_2)\ldots(\exists z_n)\,(z_1 \neq z_2\ \wedge\ z_1 \neq z_3\ \wedge\ \ldots\ \wedge\ z_{n-1} \neq z_n)\ . \tag{1}$$

(Here the expression is meant to contain all the inequalities $z_i \neq z_j$ for $1 \leq i < j \leq n$, so that it "says" that there exist $z_1, z_2, \ldots, z_n$ which are all different.) Applying the choice rule n times, as just described, we may obtain

$$y_1 \neq y_2\ \wedge\ y_1 \neq y_3\ \wedge\ \ldots\ \wedge\ y_{n-1} \neq y_n \tag{2}$$

Now read on ...

Proof. Suppose that the theory $\mathbf{A}$ has arbitrarily large finite models which respect equality. Since it has models, it is consistent.

From this form a new theory $\mathbf{A}'$ by adding a countable number of new constant symbols $b_1, b_2, \ldots$ (and extending the axiom schemata of predicate logic as necessary to encompass the new symbols). Then $\mathbf{A}'$ is consistent by Lemma B.5. ▶ B.5

Now, for each natural number n, extend the theory $\mathbf{A}'$ by adding new axioms

$$b_i \neq b_j \quad \text{for all } i, j \text{ such that } 1 \leq i < j \leq n\ .$$

(These axioms "say" that $b_1, b_2, \ldots, b_n$ are all different.) Call the theory with these new axioms $\mathbf{B}_n$.

We now prove that $\mathbf{B}_n$ is consistent for each n. Assume not. Then there is an expression P in $\mathbf{B}_n$ such that $\vdash P \wedge \neg P$ in $\mathbf{B}_n$. Choose a proof of this,

$$L_1, L_2, \ldots, L_k, \quad \text{say, where } L_k \text{ is } P \wedge \neg P\,. \tag{3}$$

Now let $y_1, y_2, \ldots, y_n$ be new variable symbols which occur nowhere in the proof (3). Create a new "proof" by replacing $b_1, b_2, \ldots, b_n$ wherever they appear in (3) by $y_1, y_2, \ldots, y_n$ (respectively); call it

$$L'_1, L'_2, \ldots, L'_k, \quad \text{where } L'_k \text{ is } P' \wedge \neg P', . \tag{4}$$

Now, at the beginning of this, add lines $K_0, K_1, \ldots, K_n$, where K_0 and K_n are the expressions (1) and (2) above, and $K_1, K_2, \ldots, K_{n-1}$ are the intermediate steps necessary to prove (2), as outlined in the preliminary remarks. Next, add the statement

$$(\exists y_1)(\exists y_2)\ldots(\exists y_n)(P \wedge \neg P) \tag{5}$$

at the end of the new proof. Calling this last statement M, then, our new proof is:

$$K_0, K_1, \ldots K_n, L'_1, L'_2, \ldots, L'_k, M\,.$$

This is then a proof (with choice) of the deduction

$$\text{Assertion (1)} \vdash (\exists y_1)(\exists y_2)\ldots(\exists y_n)(P' \wedge \neg P')\ .$$

To see this, observe that the first line K_0 is just the hypothesis, the next few lines $K_1, K_2, \ldots, K_n$ are valid proof steps, as described in the preliminary remarks, and the last line M follows from L'_k by ordinary predicate logic. It remains to check the lines $L'_1, L'_2, \ldots, L'_k$. Each line L'_r comes from replacing each b_i by y_i in the line L_r of the original proof of $P \wedge \neg P$. Where this line is justified by Modus Ponens or Universal Generalisation in the old proof, it still is in the new one. If the line L_r is an axiom of $\mathbf{A}'$ which is not a schema, then it is also an axiom of $\mathbf{A}$ and so is unchanged in the new proof, since it cannot contain any of the constants b_i. If it is an instance of an axiom schema of $\mathbf{A}$, then the original schema could not have mentioned any of the constants b_i, and so the line in the new proof is another instance of the same schema. Finally, if the line L_r is one of the new axioms $b_i \neq b_j$, then the corresponding line L'_r is $y_i \neq y_j$ and this follows from (2), which is line K_n.

By our original assumption, there is a model M of $\mathbf{A}$ with at least n members, and then (1) is true in this model. But $(\exists y_1)(\exists y_2)\ldots(\exists y_n)(P' \wedge \neg P')$ is self-contradictory and so false in any model. This means that $\mathbf{A}$ is inconsistent. This was proved on the assumption that $\mathbf{B}_n$ was inconsistent. Thus we conclude that $\mathbf{B}_n$ is consistent for every n.

Now consider the theory $\mathbf{B}_\infty$ which we manufacture the same way, except that we add a countably infinite set of new constants $b_1, b_2, \ldots$ and axioms saying that they are all unequal. Then every finite subset of the axioms of $\mathbf{B}_\infty$ must be contained in the axioms of one of our $\mathbf{B}_n$, and so be consistent and therefore have a model. By the Compactness Theorem then
▶ B.12 $\mathbf{B}_\infty$ has a model also. By Theorem B.12, it has a model respecting equality, and any such model must be infinite because in it, all the b_i must be interpreted as distinct elements. ■

C.6 Remark

▶ B.13 This remark applies only to systems with equality. Remark B.13 above tells us that, for any natural number n we care to choose, we can give a finite set of axioms which in effect together say "The model has exactly n members". (This requires adding n new constant symbols.) In fact, for any given n, we can do this with just one axiom, by simply "anding"

B.13 the axioms given in B.13 together. For example, to say that the model has exactly three members, use

$$\neg(a = b) \ \wedge \ \neg(a = c) \ \wedge \ \neg(b = c) \ \wedge \ (\forall x)(\,(x = a) \vee (x = b) \vee (x = c)\,)$$

If a theory has this axiom and no more, then any model of it which respects equality must have exactly n members. If we add this axiom to an existing theory, then either the resulting theory will be inconsistent or any model will have exactly n members.

Let F_n be the axiom above which states that the model has exactly n members. Then $\neg F_n$ states that the model does not have n members, and so the countably infinite set of axioms

$$\neg F_0, \neg F_1, \neg F_2, \ldots,$$

taken together, say "the model is infinite".

We can also force a model to be infinite with only one axiom if we add a new binary function symbol, f say. Use the axiom

$$(\forall x)(\forall y)(\forall u)(\forall v)(f(x, y) = f(u, v) \ \Rightarrow \ x = u \ \wedge \ y = v)\,.$$

This simply states that f is injective. But that makes f_M an injective function $M \times M \to M$ and the only way this can happen (since M is nonempty by definition) is for M to be infinite.

On the other hand, the last theorem above tells us that there is no set of axioms (finite or infinite and irrespective of what relations or functions we allow ourselves) which say "the model is finite" in the same way.

8. COMPUTABILITY

A Partial recursive functions

A.1 Notation

We deal with *partial functions* $\mathbb{N}^n \to \mathbb{N}$, that is, functions $D \to \mathbb{N}$, where D is a subset of $\mathbb{N}^n$. The subset D may be equal to $\mathbb{N}$, in which case the function is *total*, or it may be empty.

By *substitution* of such functions, we mean the process of obtaining, from a partial function $f : \mathbb{N}^m \to \mathbb{N}$ and partial functions $g_1, g_2, \ldots, g_m : \mathbb{N}^n \to \mathbb{N}$ the new partial function $h : \mathbb{N}^n \to \mathbb{N}$ defined by

$$h(\boldsymbol{x}) = f(g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_m(\boldsymbol{x})) \ .$$

(This is a kind of composition operation.) It is assumed that h is defined wherever the right hand side of this equation makes sense — that is, the domain of h is the intersecton of the domains of all the g_i.

Now let g be a partial function $\mathbb{N}^{n+1} \to \mathbb{N}$. The operation of *minimalisation* applied to g yields a new partial function $\mathbb{N}^n \to \mathbb{N}$, often written μg, and defined

$$\mu g(\boldsymbol{y}) = \text{the smallest value of } x \text{ such that } g(x, \boldsymbol{y}) = 0 \text{ and } g(\xi, \boldsymbol{y}) \text{ is defined for all } \xi < x,$$
$$\text{provided such an } x \text{ exists,}$$
$$\text{and is undefined if no such } x \text{ exists.}$$

This is also often written

$$\mu g(\boldsymbol{y}) = \min_x \{g(x, \boldsymbol{y}) = 0\}.$$

(Be careful of this notation; it can be misleading because it obscures the important fact that, for x to be the value of $\mu g(\boldsymbol{y})$, not only must x be the first zero of $g(x, \boldsymbol{y})$, but all preceding values must exist.)

Natural subtraction is the operation $\dot{-}$ defined

$$x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y, \\ 0 & \text{otherwise.} \end{cases}$$

A.2 Definition

A *partial recursive function* is a partial function $\mathbb{N}^n \to \mathbb{N}$ which can be obtained by a finite number of applications of the operations of substitution and minimalisation from the list:—

(i) $x \mapsto x + 1$ the successor function $\text{suc} : \mathbb{N} \to \mathbb{N}$

(ii) $\langle x_1, x_2, \ldots, x_n \rangle \mapsto x_i$ the projection function $\pi_{n,i} : \mathbb{N}^n \to \mathbb{N}$

(iii) $\langle x, y\rangle \mapsto x + y$ addition $\mathbb{N}^2 \to \mathbb{N}$

(iv) $\langle x, y\rangle \mapsto x \dot{-} y$ natural subtraction $\mathbb{N}^2 \to \mathbb{N}$

(v) $\langle x, y\rangle \mapsto xy$ multiplication $\mathbb{N}^2 \to \mathbb{N}$.

A function is *recursive* if it is both partial recursive and total.

A.3 Examples: of partial recursive and recursive functions

(i) The basic functions mentioned in Parts (i) to (v) of the definition above are all recursive. (Truly partial ones arise by application of minimalisation.)

(ii) The identity function $\mathrm{id}_{\mathbb{N}}$ on $\mathbb{N}$ is just the projection function $\pi_{1,1}$.

(iii) The zero constant function $\mathbf{0} : \mathbb{N}^0 \to \mathbb{N}$ (which takes the empty sequence $\langle\ \rangle$ to 0) is obtained from the identity function $\mathrm{id} : \mathbb{N} \to \mathbb{N}$ by minimalisation (the first x such that $\mathrm{id}(x) = 0$ is 0).

The zero constant function $\mathbb{N}^n \to \mathbb{N}$, for any n, can be obtained by minimalisation of $\pi_{n+1,1} : \mathbb{N}^{n+1} \to \mathbb{N}$. If $n \neq 0$ it can be obtained alternatively as $\pi_{n,1} \dot{-} \pi_{n,1}$.

Other constant functions $\mathbb{N}^n \to \mathbb{N}$ are obtained by composing the appropriate zero function with the successor function a suitable number of times. For example, the function $\mathbb{N}^n \to \mathbb{N}$ which takes every $\boldsymbol{x}$ to 2 is given by $\mathrm{suc}(\mathrm{suc}(\mathbf{0}(\boldsymbol{x})))$.

(iv) The *difference* function $\langle x, y\rangle \mapsto |x - y|$ is obtained by $|x - y| = (x \dot{-} y) + (y \dot{-} x)$.

(v) The *zero test* function $\mathbb{N} \to \mathbb{N}$ and the *equality test* function $\mathbb{N}^2 \to \mathbb{N}$, defined

$$\mathrm{zero}(x) = \begin{cases} 0 & \text{if } x = 0, \\ 1 & \text{otherwise} \end{cases} \qquad \mathrm{eq}(x, y) = \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{otherwise} \end{cases}$$

are obtained as $\mathrm{zero}(x) = 1 \dot{-} (1 \dot{-} x)$ and $\mathrm{eq}(x, y) = 1 \dot{-} (1 \dot{-} (|x - y|))$. And of course we have a *nonzero test* function $\mathbb{N} \to \mathbb{N}$ and the *nonequality test* function $\mathbb{N}^2 \to \mathbb{N}$, defined

$$\mathrm{nonzero}(x) = \begin{cases} 0 & \text{if } x \neq 0, \\ 1 & \text{otherwise} \end{cases} \qquad \mathrm{neq}(x, y) = \begin{cases} 0 & \text{if } x \neq y, \\ 1 & \text{otherwise} \end{cases}$$

obtained as $1 \dot{-} x$ and $1 \dot{-} |x - y|$.

In the same way we can define tests for various inequalities:

$$\mathrm{less}(x, y) = \begin{cases} 0 & \text{if } x < y, \\ 1 & \text{otherwise} \end{cases} \qquad \mathrm{gtr}(x, y) = \begin{cases} 0 & \text{if } x > y, \\ 1 & \text{otherwise} \end{cases}$$

$$\mathrm{leq}(x, y) = \begin{cases} 0 & \text{if } x \leq y, \\ 1 & \text{otherwise} \end{cases} \qquad \mathrm{geq}(x, y) = \begin{cases} 0 & \text{if } x \geq y, \\ 1 & \text{otherwise} \end{cases}$$

obtained as $\mathrm{less}(x, y) = \mathrm{nonzero}(y \dot{-} x)$, $\mathrm{gtr}(x, y) = \mathrm{nonzero}(x \dot{-} x)$, $\mathrm{leq}(x, y) = \mathrm{zero}(x \dot{-} y)$ and $\mathrm{geq}(x, y) = \mathrm{zero}(y \dot{-} x)$.

(vi) The squaring function $x \mapsto x^2$ is obtained by substituting the identity function into multiplication: mult(id(x),id(x)). Higher powers $x \mapsto x^k$ (for fixed k) are obtained by repeating this construction in the obvious way. We cannot get general exponentiation $\langle x, y\rangle \mapsto x^y$ as easily as this. It requires, dare I say it, a higher powered method.

(vii) An *integer square root*, $x \mapsto \lfloor\sqrt{x}\rfloor$ (the largest natural number $\leq \sqrt{x}$), can be described as that natural number y for which $y \leq \sqrt{x} < y+1$, which is the same as $y^2 \leq x < (y+1)^2$. It is thus the smallest natural number y for which $(y+1)^2 > x$; thus

$$\begin{aligned}\lfloor\sqrt{x}\rfloor &= \min_y\{(y+1)^2 > x\} \\ &= \min_y\{\mathrm{gtr}((y+1)^2, x) = 0\}\ .\end{aligned}$$

(viii) The function $\langle x, y\rangle \mapsto \lfloor x/y\rfloor$, defined as:

$$\lfloor x/y\rfloor = \begin{cases}\text{the greatest natural number } \leq \frac{x}{y} & \text{if } y \neq 0,\\ \text{undefined} & \text{if } y = 0.\end{cases}$$

is the natural number z such that $z \leq \frac{x}{y} < z+1$ (provided $y \neq 0$), that is, such that $yz \leq x < y(z+1)$. It is thus the smallest vaue of z for which $y(z+1) > x$, so

$$\begin{aligned}\lfloor x/y\rfloor &= \min_z\{y(z+1) > x\} \\ &= \min_z\{\mathrm{gtr}(y(z+1), x) = 0\}\ .\end{aligned}$$

(ix) The function

$$\mathrm{Rem}(x, y) = \begin{cases}\text{the remainder upon dividing } x \text{ by } y & \text{if } y \neq 0,\\ \text{undefined} & \text{if } y = 0\end{cases}$$

is obtained by $\mathrm{Rem}(x, y) = x - y\lfloor x/y\rfloor = x \dot{-} y\lfloor x/y\rfloor$ since $x \geq y\lfloor x/y\rfloor$ always.

Remark Integer division and the Remainder function are not recursive as defined, only partial recursive (both being undefined when $y = 0$). However, as we will be using them they usually give rise to recursive functions.

Suppose we have recursive functions f and g and define a new one h by

$$h(x) = \lfloor f(x)/g(x)\rfloor.$$

This clearly defines h as a partial recursive function. However, if we happen to know that $g(x)$ is never zero, then h is also a total function and so, in fact, recursive. The same idea holds good for the Remainder function.

A.4 A small extension

A partial function $f : \mathbb{N}^m \to \mathbb{N}^n$ can be written in the form $f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$, where $f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})$ are partial functions $\mathbb{N}^m \to \mathbb{N}$. We will abbreviate this $f =$

$(f_1, f_2, \ldots, f_n)$. We want to be able to define such a function to be *partial recursive* if $f_1, f_2, \ldots, f_n$ are all partial recursive, but there is a pesky detail which has to be got out of the way first.

If f is total, then its components $f_1, f_2, \ldots, f_n$ are uniquely defined by f. If f is not total then this is not necessarily the case, unless we make further restrictions. To see this: suppose we start by choosing the functions $f_1, f_2, \ldots, f_n$, to be all partial functions , but with different domains. Then defining $f = (f_1, f_2, \ldots, f_n)$, the domain of f is the intersection of the domains of $f_1, f_2, \ldots, f_n$. Consequently we can replace each f_i by its restriction to the domain of f without changing f.

It is clear now that, if we start with f, then we should *define* its components to be the functions $f_1, f_2, \ldots, f_n : \mathbb{N}^m \to \mathbb{N}$ such that $f = (f_1, f_2, \ldots, f_n)$ and which all have the same domain as f. Then these components are uniquely defined by f.

We now **define**: a partial function $f : \mathbb{N}^m \to \mathbb{N}^n$ is *partial recursive* if all its components $f_1, f_2, \ldots, f_n : \mathbb{N}^m \to \mathbb{N}$ are partial recursive (in the sense of our original definition). A function $f : \mathbb{N}^m \to \mathbb{N}^n$ is recursive if it is both partial recursive and total (or, equivalently, if all its components are recursive).

A.5 Lemma

Let $f_1, f_2, \ldots, f_n$ be partial recursive functions $\mathbb{N}^m \to \mathbb{N}$. Then the function $g : \mathbb{N}^m \to \mathbb{N}^n$ defined by $f = (f_1, f_2, \ldots, f_n)$ is partial recursive also. (The same result holds trivially for recursive functions).

[The point here is that the functions $f_1, f_2, \ldots, f_n$ may not all have the same domain and in this case they are not the components of g.]

Proof. Firstly, note that $\mathrm{dom}(g) = \mathrm{dom}(f_1) \cap \mathrm{dom}(f_2) \cap \ldots \cap \mathrm{dom}(f_n)$. Then $g_1, g_2, \ldots, g_n$, the genuine components of g, are the restrictions of the f_i to $\mathrm{dom}(g)$. For each $i = 1, 2, \ldots, n$ let us write d_i for the function $\mathbb{N}^m \to \mathbb{N}$ given by

$$d_i(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in \mathrm{dom}(f_i), \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Now $d_i(\boldsymbol{x}) = f_i(\boldsymbol{x}) \dot{-} f_i(\boldsymbol{x}) + 1$, so these functions are all partial recursive. Therefore so is their product $d(\boldsymbol{x}) = d_1(\boldsymbol{x}) d_2(\boldsymbol{x}) \ldots d_n(\boldsymbol{x})$, and this is the function

$$d(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in \mathrm{dom}(g), \\ \text{undefined} & \text{otherwise.} \end{cases}$$

But now the components of g are given by $g_i(\boldsymbol{x}) = d(\boldsymbol{x}) f_i(\boldsymbol{x})$ for each i, so these are partial recursive, as required. ■

Observe that the relation "closure under the operation of substitution" in our original definition of partial recursive functions may be replaced by closure under the two slightly simpler operations:

(a) *Presubstitution*: given partial functions $f_1, f_2, \ldots, f_n$, obtain $f = (f_1, f_2, \ldots, f_n)$.

(b) Ordinary composition.

A.6 Lemma: Pairs

There is a recursive function $P : \mathbb{N}^2 \to \mathbb{N}$ which has a recursive inverse.

(We will denote this inverse $x \mapsto (L(x), R(x))$.)

Proof. We must define the two functions, show that they are inverses and are recursive. We define P by the usual way of enumerating $\mathbb{N}$:

y \ x	0	1	2	3	
0	0	2	5	9	
1	1	4	8		
2	3	7			
3	6				etc.

It is not difficult to see that

$$P(x, y) = \frac{1}{2}(x + y)(x + y + 1) + x \tag{1}$$

and, since $\frac{1}{2}(x + y)(x + y + 1)$ must be a natural number, we have

$$P(x, y) = \left\lfloor \frac{(x + y)(x + y + 1)}{2} \right\rfloor + x$$

and so P is recursive (see the remark at the end of Lemma A.3.) ▶ A.3

Suppose now that $P(x, y) = z$ so that $x = L(z)$ and $y = R(z)$. By simple algebraic manipulations we have

From (1) $\quad (x + y)(x + y + 1) + 2x = 2z$

that is $\quad (x + y)^2 + 3x + y = 2z \tag{2}$

so $\quad (2x + 2y + 1)^2 + 8x = 8z + 1$

and so $\quad (2x + 2y + 1)^2 \leq 8z + 1 < (2x + 2y + 3)^2.$

Then $\quad 2x + 2y + 1 \leq \sqrt{8z + 1} < 2x + 2y + 3$

so $\quad \lfloor \sqrt{8z + 1} \rfloor = 2x + 2y + 1 \quad \text{or} \quad 2x + 2y + 2$

and so $\quad \lfloor \frac{\lfloor \sqrt{8z+1} \rfloor + 1}{2} \rfloor = x + y + 1.$

Thus $\quad \lfloor \frac{\lfloor \sqrt{8z+1} \rfloor + 1}{2} \rfloor - 1 = x + y$

and, since the right hand side here must be ≥ 0, we have

$$x + y = \lfloor \frac{\lfloor \sqrt{8z+1} \rfloor + 1}{2} \rfloor \dot{-} 1 = Q_1(z) \quad \text{say.} \tag{3}$$

We see that Q_1 is recursive. Substituting back into (2),

$$3x + y = 2z \dot{-} Q_1(z)^2 = Q_2(z) \quad \text{say,}$$

another recursive function. We solve the last two equations in the usual way to get $x = (Q_2(z) - Q_1(z))/2$. Since x is a natural number we must have $Q_2(z) - Q_1(z) \geq 0$ and even. Thus $x = \lfloor (Q_2(z) \dot{-} Q_1(z))/2 \rfloor$. Since this is true for all x, it gives us a formula for L as a recursive function:

$$L(z) = \left\lfloor \frac{Q_2(z) \dot{-} Q_1(z)}{2} \right\rfloor.$$

Substituting back into (3) gives $R(z) = Q_1(z) \dot{-} L(z)$. ■

Note for later use: if z and z' are two natural numbers such that $L(z) = L(z')$ and $R(z) = R(z')$, then $z = z'$.

Also we have

$$P(L(z), R(z)) = z \quad \text{for all } z$$

and

$$L(P(x, y)) = x \quad \text{and} \quad R(P(x, y)) = y \quad \text{for all } x \text{ and } y.$$

A.7 Theorem: Sequences

There is a recursive function $T : \mathbb{N}^2 \to \mathbb{N}$ such that, for any finite sequence $a_0, a_1, \ldots, a_n$ of natural numbers, there is a number w such that

$$T(w, i) = a_i \qquad \text{for } i = 0, 1, \ldots, n.$$

A.8 Lemma

Let ν be any natural number which is divisible by all of $1, 2, ..., n$. Then the $n + 1$ numbers

$$1 + \nu(i + 1) \qquad i = 0, 1, 2, \ldots, n$$

are pairwise coprime.

Proof. Any divisor of $1 + \nu(i + 1)$, other than 1, must be greater than n. Now suppose that $0 \leq i < j \leq n$ and d divides both $1 + \nu(i + 1)$ and $1 + \nu(j + 1)$. Then d divides $(j + 1)(1 + \nu(i + 1)) - (i + 1)(1 + \nu(j + 1)) = j - i$. Thus $d \leq j - i < n$ so $d = 1$. ■

A.9 The Chinese Remainder Theorem

This is a standard result in elementary number theory and will not be proved here. It states:

Let $m_0, m_1, \ldots, m_k$ be natural numbers which are pairwise coprime and let $a_0, a_1, \ldots, a_k$ be any natural numbers. Then there is a natural number x such that

$$x \equiv a_i \mod m_i \qquad \text{for } i = 0, 1, \ldots, k \ .$$

We need to make a slight modification:

Let $m_0, m_1, \ldots, m_k$ be nonzero natural numbers which are pairwise coprime and let $a_0, a_1, \ldots, a_k$ be any natural numbers. Then there is a nonzero natural number x such that

$$x \equiv a_i \mod m_i \qquad \text{for } i = 0, 1, \ldots, k \ .$$

To see this, suppose that x should be zero in the original version (which can only happen if all the a_i are zero). Then we may replace x by the product $m_0 m_1 \ldots m_k$ and the result is still true.

A.10 Lemma

Let $a_0, a_1, \ldots, a_n$ be a finite sequence of natural numbers. Then there are natural numbers u and v such that

$$\mathrm{Rem}(u + 1, 1 + v(i + 1)) = a_i \qquad \text{for } i = 0, 1, \ldots, n \ .$$

Proof. Let $A = \max\{a_0, a_1, \ldots, a_n\}$ and $v = (A+1)n!$. By Lemma A.8 above, the numbers ▶ A.8
$1 + v(i + 1)$ are pairwise coprime. Also $a_i < v < 1 + v(i + 1)$ for all i. By the modified version of the Chinese Remainder Theorem, there is a nonzero natural number x such that

$$x \equiv a_i \mod 1 + v(i + 1) \qquad \text{for } i = 0, 1, \ldots, n$$

and so, setting $x = u + 1$, there is a natural number u such that

$$u + 1 \equiv a_i \mod 1 + v(i + 1) \qquad \text{for } i = 0, 1, \ldots, n \, ,$$

that is

$$\begin{aligned} \mathrm{Rem}(u + 1, 1 + v(i + 1)) &= \mathrm{Rem}(a_i, 1 + v(i + 1)) \\ &= a_i \qquad\qquad \text{since } a_i < 1 + v(i + 1). \blacksquare \end{aligned}$$

Proof of the theorem. Define

$$T(w, i) = \mathrm{Rem}(L(w) + 1,\ 1 + R(w)(i + 1)).$$

Then T is obviously recursive (see the remark at the end of Lemma A.3.) Now, given ▶ A.3
$a_0, a_1, \ldots, a_n$, by the previous lemma, there are integers u and v such that $a_i = \mathrm{Rem}(u + 1, 1 + v(i + 1))$ for $i = 0, 1, \ldots, n$. Setting $w = P(u, v)$ does the trick. ■

A.11 Remarks

It would be nice if we could construct any recursive function as in Definition A.2, but in such a way that all intermediate functions involved are also recursive, i.e. total. But we observe that the definition allows the construction of a recursive function in such a way that one or more of the intermediate functions involved are in fact partial; it is by no means obvious that such a construction can be modified to avoid these intermediate partial functions.

► A.2

A *regular* function is a total function which yields a total function when minimalised. More precisely, a function $f : \mathbb{N}^{n+1} \to \mathbb{N}$ is regular if it is total and, for every $\boldsymbol{y} \in \mathbb{N}^n$, there is some $x \in \mathbb{N}$ such that $f(x, \boldsymbol{y}) = 0$.

► A.2

To return to our wishful thinking: If in Definition A.2 we restrict the operation of minimalisation to apply only to regular functions, then all functions so produced would be total and so recursive. It would be nice if all recursive functions could be formed in this way, but, as stated above, it is not at all obvious. Nevertheless it is so. We will prove this as a corollary to a rather big theorem later in this Chapter (Corollary C.6).

► C.6

A.12 Definition: Recursive sets and predicates

A subset A of $\mathbb{N}^n$ is *recursive* if its characteristic function $\chi_A : \mathbb{N}^n \to \mathbb{N}$ is recursive. Here as usual

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}$$

An n-ary predicate P (predicate on $\mathbb{N}^n$) is *recursive* if its *boolean function* $\bar{P}$ is recursive. Here as usual the boolean function is

$$\bar{P}(x) = \begin{cases} 1 & \text{if } P(x) \text{ is true} \\ 0 & \text{if } P(x) \text{ is false.} \end{cases}$$

It follows that a subset A of $\mathbb{N}^n$ is recursive if and only if the predicate "$x \in A$" is recursive. Similarly a predicate P on $\mathbb{N}^n$ is recursive if and only if the subset $\{\, x : x \in \mathbb{N}^n \text{ and } P(x) \,\}$ is recursive.

A.13 Lemma

(i) Let P and Q be two n-ary predicates (defined on $\mathbb{N}^n$). If P and Q are both recursive then so are $\neg P$, $P \wedge Q$, $P \vee Q$, $P \Rightarrow Q$ and $P \Leftrightarrow Q$.

Further, if $f_1, f_2, \ldots, f_n$ are recursive functions $\mathbb{N}^m \to \mathbb{N}$, then $P(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$ is an m-ary recursive predicate.

(ii) Let P be a unary predicate (on $\mathbb{N}$). If P is recursive, then so are the unary predicates

$$(\forall \xi \le x)P(\xi) \quad , \quad (\exists \xi \le x)P(\xi) \quad , \quad (\forall \xi < x)P(\xi) \quad \text{and} \quad (\exists \xi < x)P(\xi)$$

More generally, suppose P is a predicate on $\mathbb{N}^{n+1}$. If P is recursive, then so are the $(n{+}1)$-ary predicates

$$(\forall \xi \le x)P(\xi, \boldsymbol{y}) \quad , \quad (\exists \xi \le x)P(\xi, \boldsymbol{y}) \quad , \quad (\forall \xi < x)P(\xi, \boldsymbol{y}) \quad \text{and} \quad (\exists \xi < x)P(\xi, \boldsymbol{y}).$$

(iii) If f and g are recursive functions $\mathbb{N} \to \mathbb{N}$, then the n-ary predicates

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) \quad , \quad f(\boldsymbol{x}) \neq g(\boldsymbol{x}) \quad , \quad f(\boldsymbol{x}) \leq g(\boldsymbol{x}) \quad \text{and} \quad f(\boldsymbol{x}) < g(\boldsymbol{x})$$

are all recursive also.

(iv) If $P(x, \boldsymbol{y})$ is an $(n+1)$-ary recursive predicate, then the function $f : \mathbb{N}^n \to \mathbb{N}$ given by

$$f(\boldsymbol{y}) \;=\; \min_x\{P(x, \boldsymbol{y})\}$$

is partial recursive.

Proof. (i) The boolean functions $\bar{P}$ and $\bar{Q} : \mathbb{N}^n \to \mathbb{N}$ are recursive. Now

$$\begin{aligned} \overline{\neg P}(x) \;&=\; 1 \dot{-} \bar{P}(x) \qquad \text{and} \\ \overline{P \wedge Q}(x) \;&=\; \bar{P}(x).\bar{Q}(x). \end{aligned}$$

That $P \vee Q$, $P \Rightarrow Q$ and $P \Leftrightarrow Q$ are recursive now follows immediately.

The boolean function of $P(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$ is $\bar{P}(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$.

(iii) The boolean function of the predicate $f(\boldsymbol{x}) = g(\boldsymbol{x})$ is $\mathrm{eq}(f(\boldsymbol{x}), g(\boldsymbol{x}))$ and, of course, $f(\boldsymbol{x}) \neq g(\boldsymbol{x})$ is $\neg(f(\boldsymbol{x}) = g(\boldsymbol{x}))$.

(iv) $f(\boldsymbol{y}) \;=\; \min_x\{\, 1 \dot{-} \bar{P}(x, \boldsymbol{y}) = 0 \,\}$, which is a straight minimalisation.

We note, for use in the next part of the proof, that if P is such that, for every $\boldsymbol{y} \in \mathbb{N}^n$, there is some x such that $P(x, \boldsymbol{y})$, then f is recursive.

(ii) The boolean function $\bar{P} : \mathbb{N}^n \to \mathbb{N}$ is recursive. Notice that $(\forall \xi < x) P(x, \boldsymbol{y})$ if and only if

$$\min_\xi\{\, \bar{P}(\xi, \boldsymbol{y}) = 0 \text{ or } \xi = x \,\} \;=\; x\,.$$

Note also that the predicate being minimised here is recursive (by (i) and the first half of (iii)). Thus (usng the note at the end of the proof of (iv) above), the left hand side of this equation is a recursive function. Then the whole equation (as a predicate) is recursive, by (iii) again.

The other three results now follow immediately.

(Second half) The boolean function of the predicate $x < y$ is

$$(\exists \xi < y)(\, x + \xi + 1 = y)$$

and $x \leq y$ is equivalent to $\neg(y < x)$, so these predicates are recursive. Substitute the functions f and g in these (using (i)) to see that $f(\boldsymbol{x}) < g(\boldsymbol{x})$ and $f(\boldsymbol{x}) \leq g(\boldsymbol{x})$ are recursive also. ■

B Primitive recursive functions

B.1 Definition

Given total functions $g : \mathbb{N}^n \to \mathbb{N}$ and $h : \mathbb{N}^{n+2} \to \mathbb{N}$, the operation of *primitive recursion* obtains the (total) function $f : \mathbb{N}^{n+1} \to \mathbb{N}$ given by

$$f(0, \boldsymbol{y}) = g(\boldsymbol{y}) \tag{1A}$$
$$f(x+1, \boldsymbol{y}) = h(x, f(x, \boldsymbol{y}), \boldsymbol{y}) \qquad \text{for all } x \in \mathbb{N}. \tag{1B}$$

Remark The function f so defined exists, is total and is uniquely defined by g and h. This has already been discussed in the Set Theory chapter.

B.2 Theorem

With the notation of the above definition, if g and h are recursive functions, then so is f.

▶ A.7

Proof. Let T be the function defined in Theorem A.7. Then, for any x and $\boldsymbol{y}$, there is a w so that

$$T(w, \xi) = f(\xi, \boldsymbol{y}) \qquad \text{for all } \xi = 0, 1, \ldots, x. \tag{2}$$

Then, substituting in Equations (1A) and (1B) above,

$$T(w, 0) = g(\boldsymbol{y}) \tag{3A}$$
$$\text{and} \qquad T(w, \xi+1) = h(\xi, T(w, \xi), \boldsymbol{y}) \qquad \text{for} \quad \xi = 0, 1, \ldots, x-1\ . \tag{3B}$$

Also conversely, if w is any natural number such that Equations (3A) and (3B) hold, then $T(w, \xi) = f(\xi, \boldsymbol{y})$ for all $\xi = 0, 1, \ldots, x$.

Now, for any x and $\boldsymbol{y}$, let w_0 be the least w satisfying these equations. Then, for every x and $\boldsymbol{y}$, w_0 is uniquely defined, so we may consider it to be a function of x and $\boldsymbol{y}$ and write it as $w_0(x, \boldsymbol{y})$. We have in fact defined it as

$$w_0(x, \boldsymbol{y}) = \min_w \{\, T(w, 0) = g(\boldsymbol{y}) \quad \text{and} \quad (\forall \xi < x)\big(T(w, \xi+1) = h(\xi, T(w, \xi), \boldsymbol{y})\big) \,\}\ .$$

and we know that Equation (2) holds for it in the sense that , for any x and $\boldsymbol{y}$,

$$T(w_0(x, \boldsymbol{y}), \xi) = f(\xi, \boldsymbol{y}) \quad \text{for all } \xi = 0, 1, \ldots, x$$

and, in particular, that

$$T(w_0(x, \boldsymbol{y}), x) = f(x, \boldsymbol{y})\,. \tag{4}$$

▶ A.13

Since T is recursive, it follows from Lemma A.13 that so is w_0 and then from Equation (4) that so is f. ■

B.3 Definition

A function $f : \mathbb{N}^n \to \mathbb{N}$ is *primitive recursive* if it can be obtained by a finite number of applications of the operations of substitution and primitive recursion from the functions in the following list:—

(i) $x \mapsto x + 1$, the successor function $\mathrm{suc} : \mathbb{N} \to \mathbb{N}$,

(ii) $\mathbf{0}$ the constant zero function $\mathbb{N}^0 \to \mathbb{N}$,

(iii) $(x_1, x_2, \ldots, x_n) \mapsto x_i$ the projection function $\pi_{n,i} : \mathbb{N}^n \to \mathbb{N}$.

From the last theorem we see that every primitive recursive function is recursive. On the other hand, there are recursive functions which are not primitive recursive: an example will be given later in this chapter (Section D). ▶ D

B.4 Examples of primitive recursive functions

(i) The identity function $\mathrm{id} : \mathbb{N} \to \mathbb{N}$ is still the projection function $\pi_{1,1}$.

(ii) Addition $\mathbb{N}^2 \to \mathbb{N}$:

$$x + 0 = x \qquad (\mathrm{add}(x, 0) = \mathrm{id}(x))$$

$$x + (y + 1) = (x + y) + 1 \qquad (\mathrm{add}(x, y + 1) = \mathrm{suc}(\mathrm{add}(x + y)))$$

(iii) The constant function $x \mapsto 0$ (as a function $\mathbb{N} \to \mathbb{N}$) is obtained from $\mathbf{0}$ = the constant 0, as a function $\mathbb{N}^0 \to \mathbb{N}$, and $\pi_{2,2}$ by primitive recursion.

The constant zero functions $\mathbb{N}^n \to \mathbb{N}$ for $n > 1$ can now be obtained by composing the constant zero function $\mathbb{N} \to \mathbb{N}$, just constructed, with the projection functions $\pi_{n,1}$.

Other constant functions (either $\mathbb{N}^0 \to \mathbb{N}$ or $\mathbb{N}^n \to \mathbb{N}$) are obtained by composing these zero functions with the successor function a suitable number of times.

(iv) Multiplication $\mathbb{N}^2 \to \mathbb{N}$:

$$x0 = 0 \qquad (\mathrm{mul}(x, 0) = \text{constant zero function } \mathbb{N} \to \mathbb{N})$$

$$x(y + 1) = (xy) + x \qquad (\mathrm{mul}(x, y + 1) = \mathrm{add}(\mathrm{mul}(xy), x))$$

(v) Natural subtraction $\mathbb{N}^2 \to \mathbb{N}$:

First note that the *natural predecessor* function $p(x) = x \dot{-} 1$ is primitive recursive, because it is given by $p(0) = 0$ and $p(x + 1) = x$. Now we can define natural subtraction by

$$x \dot{-} 0 = x$$

$$x \dot{-} (y + 1) = p(x \dot{-} y) \ .$$

B.5 Remark

From these examples we see that an alternative definition of primitive recursive functions would be to make the following changes to the definition of partial recursive functions:

replace the operation of minimalisation by that of primitive recursion and add the zero function to the basic list.

By the same token, an alternative definition of partial recursive functions is afforded by starting with the definition of primitive recursive function and adding the operation of minimalisation.

B.6 Examples continued

(vi) The factorial function $\mathbb{N} \to \mathbb{N}$:

$$0! = 1$$

$$(x+1)! = x!.(x+1) \ .$$

(vii) Exponentiation $\mathbb{N}^2 \to \mathbb{N}$:

$$x^0 = 1$$

$$x^{y+1} = x^y x \ .$$

► A.3

(viii) The functions zero, eq, nonzero, neq and $\langle x, y\rangle \mapsto |x-y|$ are defined in A.3(v) above.

B.7 Theorem

If $f : \mathbb{N}^{n+1} \to \mathbb{N}$ is recursive or primitive recursive, then so (respectively) are

$$g : \mathbb{N}^{n+1} \to \mathbb{N} \qquad \text{defined by} \qquad g(x, \boldsymbol{y}) = \sum_{i=0}^{x} f(i, \boldsymbol{y})$$

$$h : \mathbb{N}^{n+1} \to \mathbb{N} \qquad \text{defined by} \qquad h(x, \boldsymbol{y}) = \prod_{i=0}^{x} f(i, \boldsymbol{y})$$

Proof. $g(0, \boldsymbol{y}) = f(0, \boldsymbol{y})$ and $g(x+1, \boldsymbol{y}) = g(x, \boldsymbol{y}) + f(x+1, \boldsymbol{y})$,

$h(0, \boldsymbol{y}) = f(0, \boldsymbol{y})$ and $g(x+1, \boldsymbol{y}) = g(x, \boldsymbol{y}).f(x+1, \boldsymbol{y})$. ■

In the case that f is recursive, this theorem would have been awkward to prove without the help of Theorem B.2.

► B.2

B.8 Theorem

If $f : \mathbb{N}^{n+1} \to \mathbb{N}$ is a recursive or primitive recursive function, then so is the function obtained from it by *bounded minimalisation*, that is,

$$h(m, \boldsymbol{y}) = \begin{cases} \text{the least value of } x \text{ such that } x \leq m \text{ and } f(x, \boldsymbol{y}) = 0 & \text{if such an } x \text{ exists,} \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Noting that

$$\text{zero}(f(x,\boldsymbol{y})) = \begin{cases} 1 & \text{if } f(x,\boldsymbol{y}) \neq 0, \\ 0 & \text{otherwise} \end{cases}$$

we have

$$\prod_{i=0}^{x} \text{zero}(f(i,\boldsymbol{y})) = \begin{cases} 1 & \text{if } f(i,\boldsymbol{y}) \neq 0 \text{ for all } i \leq x, \\ 0 & \text{otherwise} \end{cases}$$

Also

$$(x+1).\,\text{nonzero}(f(x+1)) \;=\; \begin{cases} x+1 & \text{if } f(x+1,\boldsymbol{y}) = 0, \\ 0 & \text{otherwise} \end{cases}$$

so, multiplying the last two functions,

$$(x+1).\,\text{nonzero}(f(x+1)) \prod_{i=0}^{x} \text{zero}(f(i,\boldsymbol{y})) \\ = \begin{cases} x+1 & \text{if } f(x+1,\boldsymbol{y}) = 0 \;\text{ and }\; f(i,\boldsymbol{y}) \neq 0 \text{ for all } i \leq x, \\ 0 & \text{otherwise} \end{cases}$$

and this is primitive recursive and so, defining a function $\theta : \mathbb{N}^{n+1} \to \mathbb{N}$ by

$$\theta(0,\boldsymbol{y}) = 0$$
$$\theta(x+1,\boldsymbol{y}) = (x+1).\,\text{nonzero}(f(x+1,\boldsymbol{y})) \prod_{i=0}^{x} \text{zero}(f(i,\boldsymbol{y})),$$

θ is also primitive recursive and

$$\theta(x,\boldsymbol{y}) = \begin{cases} x & \text{if } f(x,\boldsymbol{y}) = 0 \text{ and } f(i,\boldsymbol{y}) \neq 0 \text{ for all } i < x, \\ 0 & \text{otherwise.} \end{cases}$$

So

$$h(m,\boldsymbol{y}) = \sum_{x=0}^{m} \theta(x,\boldsymbol{y}) \qquad \text{is primitive recursive also.}$$

■

Notation It will be convenient to use the notation $\min_{x \leq m}\{f(x,\boldsymbol{y}) = 0\}$ for the result of applying bounded minimalisation to the function f.

B.9 Corollary

Now we see that all the functions given as examples of recursive functions so far (the total ones only!) are in fact primitive recursive, because in each case where minimalisation was used, bounded minimalisation could have been used instead. For example, integer division and integer square root can be redefined:

$$\left\lfloor \frac{x}{y} \right\rfloor = \min_{z \leq x}\{\text{gtr}(y(z+1)\,,\, x) = 0\}$$

$$\lfloor\sqrt{x}\rfloor = \min_{z\le x}\{\mathrm{gtr}((y+1)^2\,,\,x)=0\}.$$

In the same way, the functions P, L and R used to set up a recursive bijection between $\mathbb{N}$ and $\mathbb{N}^2$ are primitive recursive.

B.10 Definition: Primitive recursive sets and predicates

(The definitions here are the same as for recursive sets and predicates.)

A subset A of $\mathbb{N}^n$ is *primitive recursive* if its characteristic function $\chi_A : \mathbb{N}^n \to \mathbb{N}$ is primitive recursive.

An n-ary predicate P (predicate on $\mathbb{N}^n$) is *primitive recursive* if its boolean function $\bar{P}$ is primitive recursive.

It follows that a subset A of $\mathbb{N}^n$ is primitive recursive if and only if the predicate "$x \in A$" is primitive recursive. Similarly a predicate P on $\mathbb{N}^n$ is primitive recursive if and only if the subset $\{\, x : x \in \mathbb{N}^n \text{ and } P(x) \,\}$ is primitive recursive.

B.11 Lemma

(i) Let P and Q be two n-ary predicates (defined on $\mathbb{N}^n$). If P and Q are both primitive recursive then so are $\neg P$, $P \wedge Q$, $P \vee Q$, $P \Rightarrow Q$ and $P \Leftrightarrow Q$.

Further, if $f_1, f_2, \ldots, f_n$ are primitive recursive functions $\mathbb{N}^m \to \mathbb{N}$, then $P(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$ is an m-ary primitive recursive predicate.

(ii) Let P be a unary predicate (on $\mathbb{N}$). If P is primitive recursive, then so are the unary predicates

$$(\forall\xi \le x)P(\xi) \quad , \quad (\exists\xi \le x)P(\xi) \quad , \quad (\forall\xi < x)P(\xi) \quad \text{and} \quad (\exists\xi < x)P(\xi)$$

More generally, suppose P is a predicate on $\mathbb{N}^{n+1}$. If P is primitive recursive, then so are the $(n{+}1)$-ary predicates

$$(\forall\xi \le x)P(\xi,\boldsymbol{y}) \quad , \quad (\exists\xi \le x)P(\xi,\boldsymbol{y}) \quad , \quad (\forall\xi < x)P(\xi,\boldsymbol{y}) \quad \text{and} \quad (\exists\xi < x)P(\xi,\boldsymbol{y}).$$

(iii) If f and g are primitive recursive functions $\mathbb{N} \to \mathbb{N}$, then the n-ary predicates

$$f(\boldsymbol{x}) = g(\boldsymbol{x}) \quad , \quad f(\boldsymbol{x}) \ne g(\boldsymbol{x}) \quad , \quad f(\boldsymbol{x}) \le g(\boldsymbol{x}) \quad \text{and} \quad f(\boldsymbol{x}) < g(\boldsymbol{x})$$

are all primitive recursive also.

Proof. The proofs of (i) and (iii) are exactly the same as the corresponding proofs of Lemma A.13. (The proof there of (ii) involves minimalisation, so cannot be used here, so ...)

► A.13

(ii) The boolean function of $(\forall\xi \le x)P(\xi)$ is

$$\prod_{\xi=0}^{x} \bar{P}(\xi,\boldsymbol{y})$$

and now the other three follow immediately. ■

B.12 Finite sequences again

It will be useful to have some primitive recursive functions relating members of $\mathbb{N}^n$ and $\mathbb{N}$, in the way the functions P, L and R relate $\mathbb{N}^2$ and $\mathbb{N}$. More specifically, for each n we want primitive recursive functions $P_n : \mathbb{N}^n \to \mathbb{N}$ and $E_{n,1}, E_{n,2}, \ldots, E_{n,n}$ all $: \mathbb{N} \to \mathbb{N}$ such that

$$E_{n,i}(P_n(x_1, x_2, \ldots, x_n)) \;=\; x_i \qquad \text{for every sequence } (x_1, x_2, \ldots, x_n) \text{ and } i = 1, 2, \ldots, n$$

$$P(E_{n,1}(w), E_{n,2}(w), \ldots, E_{n,n}(w)) \;=\; w \qquad \text{for every } w \in \mathbb{N}\,.$$

B.13 Definition

We define the functions $P_n : \mathbb{N}^n \to \mathbb{N}$ and $\boldsymbol{E}_n : \mathbb{N} \to \mathbb{N}^n$ by induction over n:

$$\begin{aligned} P_0() \;&=\; 0 \quad ; \quad P_1(x) = x \quad ; \\ P_{n+1}(x_1, x_2, \ldots, x_{n+1}) \;&=\; P(x_1, P_n(x_2, x_3, \ldots, _{n+1})) \qquad \text{for all } n \geq 2\,, \end{aligned}$$

where P is the function $\mathbb{N}^2 \to \mathbb{N}$ defined earlier, and

$$\begin{aligned} \boldsymbol{E}_0(w) = () \quad ; \quad \boldsymbol{E}_1(w) = (w) \quad ; \\ \boldsymbol{E}_{n+1}(w) \;=\; (L(w), LR(w), LR^2(w), \ldots, LR^{n-1}(w), R^n(w)) \qquad \text{for all } n \geq 2 \end{aligned}$$

so that, for example, $\boldsymbol{E}_2(w) = (L(w), R(w))$ and $\boldsymbol{E}_3(w) = (L(w), LR(w), R^2(w))$.

It will be convenient to write $E_{n,i}$ for the various components of $\boldsymbol{E}_n$, so that

$$\begin{aligned} E_{n,1}(w) \;&=\; L(w) \qquad \text{for all } n \geq 1\,, \\ E_{n,i}(w) \;&=\; LR^{i-1}(w) \qquad \text{for all } i, n \text{ such that } 2 \leq i \leq n-1\,, \\ E_{n,n} \;&=\; R^{n-1}(w) \qquad \text{for all } n \geq 1\,. \end{aligned}$$

(Here we are using a simplified notation for composites. For instance, $LR(w)$ means $L(R(w))$, $LR^2(w)$ means $L(R(R(w)))$ and so on.)

B.14 Lemma

With the notation of the last definition, the functions P_n and $\boldsymbol{E}_n$ are all primitive recursive (to say that $\boldsymbol{E}_n$ is primitive recursive means, as usual, that all its component functions $E_{n,i}$ are primitive recursive).

Furthermore, these functions are inverse bijections, that is

$$\begin{aligned} \boldsymbol{E}_n(P(\boldsymbol{x})) \;&=\; \boldsymbol{x} \qquad \text{for all } \boldsymbol{x} \in \mathbb{N}^n \text{ and} \\ P(\boldsymbol{E}(w)) \;&=\; w \qquad \text{for all } w \in \mathbb{N}\,. \end{aligned}$$

Proof. This is all easy. ■

B.15 Remarks

We now want to extend this idea to a function which acts like the P_n above, but so that one function will do for all sequences of all lengths. This function and its properties, which we will prove here, are crucial to a major result in the next section and also to the proof of Gödel's Theorem in Chapter 9.

▶ 9

We will call the function σ and to say that it is defined on all sequences of all (finite) lengths means that it is a function

$$\sigma \,:\, \bigcup_{n=0}^{\infty} \mathbb{N}^n \;\to\; \mathbb{N}\,.$$

Now to ask for such a function σ to be primitive recursive (or recursive for that matter) is problematical: we have no definition of primitive recursiveness for functions defined on this domain. It is possible to make a definition, but we do not need to go to these lengths. What we need to know is that the operations we want to perform on sequences (adding or removing a first element, getting the value of the ith element and replacing the ith element by some given value) are reflected in primitive recursive functions defined on $\sigma(\boldsymbol{x})$.

The idea is fairly simple: we can easily describe a sequence of any length by two numbers, its length, n say, and P_n of the sequence. In other words, the sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is specified by the pair $(n, P_n(\boldsymbol{x}))$. We want however to describe the sequence by a single number; but this is now easy, use the function P on this pair. So we arrive at ...

B.16 Definition

We define the function

$$\sigma \,:\, \bigcup_{n=0}^{\infty} \mathbb{N}^n \;\to\; \mathbb{N}\,.$$

as follows: for any sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$,

$$\sigma(\boldsymbol{x}) \;=\; P(n\,,\, P_n(x_1, x_2, \ldots, x_n))\;.$$

Looking at the definitions in the previous section we see that we have an alternative way of expressing this definition: $\sigma(x_1, x_2, \ldots, x_n) \;=\; P_{n+1}(n, x_1, x_2, \ldots, x_n)$.

We note that it follows immediately from this definition that the restriction σ_n of σ to any $\mathbb{N}^n$,

$$\sigma_n \,:\, \mathbb{N}^n \to \mathbb{N}$$

is primitive recursive.

B.17 Lemma

There are primitive recursive functions $\mathrm{len} : \mathbb{N} \to \mathbb{N}$, $\mathrm{ent} : \mathbb{N}^2 \to \mathbb{N}$, $\mathrm{del} : \mathbb{N} \to \mathbb{N}$, $\mathrm{add} : \mathbb{N}^2 \to \mathbb{N}$, $\mathrm{rep} : \mathbb{N}^3 \to \mathbb{N}$ with these properties:

(i) $\mathrm{len}(w)$ gets the length of the sequence: if $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $\sigma(\boldsymbol{x}) = w$ then $\mathrm{len}(w) = n$.

(ii) $\mathrm{ent}(i,w)$ gets the ith entry of the sequence: if $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, $\sigma(\boldsymbol{x}) = w$ and $1 \le i \le n$ then $\mathrm{ent}(i,w) = x_i$.

(iii) $\mathrm{del}(w)$ represents removing the first entry of the sequence: if $n \ge 1$, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, $\boldsymbol{x}' = (x_2, x_3, \ldots, x_n)$, $\sigma(\boldsymbol{x}) = w$ and $\sigma(\boldsymbol{x}') = w'$ then $\mathrm{del}(w) = w'$.

(iv) $\mathrm{add}(z,w)$ represents adding a new first entry z to the sequence: if $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, $\boldsymbol{x}' = (z, x_1, x_2, \ldots, x_n)$, $\sigma(\boldsymbol{x}) = w$ and $\sigma(\boldsymbol{x}') = w'$ then $\mathrm{add}(z,w) = w'$.

(v) $\mathrm{rep}(r,w,y)$ represents replacing the rth entry of the sequence by the value y: if $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, $\boldsymbol{x}' = (x_1, x_2, \ldots, x_{r-1}, y, x_{r+1}, \ldots, x_n)$, $\sigma(\boldsymbol{x}) = w$ and $\sigma(\boldsymbol{x}') = w'$ then $\mathrm{rep}(r,w,y) = w'$.

Proof. (i) This is trivial: $\mathrm{len}(w) = L(w)$.

(ii) Given $w = \sigma(x_1, x_2, \ldots, x_n) = P_{n+1}(n, x_1, x_2, \ldots, x_n)$ we have $\mathrm{ent}(i,w) = x_i = E_{n+1,i+1}(w)$.

(iii) Given $w = \sigma(x_1, x_2, \ldots, x_n) = P_{n+1}(n, x_1, x_2, \ldots, x_n)$, and noting that then $P_{n-1}(x_2, x_3, \ldots, x_n) = R^2(w)$, we have

$$\mathrm{del}(w) \;=\; \sigma(x_2, x_3, \ldots, x_n) \;=\; P_n(n-1, x_2, x_3, \ldots, x_n) \;=\; P(L(w)-1\,,\, R^2(w))\ .$$

(iv) Given $w = \sigma(x_1, x_2, \ldots, x_n) = P_{n+1}(n, x_1, x_2, \ldots, x_n)$,

$$\begin{aligned}
\mathrm{add}(z,w) &= \sigma(z, x_1, x_2, \ldots, x_n) \\
&= P(n+1, P_{n+1}(z, x_1, x_2, \ldots, x_n) \\
&= P(n+1, P(z, P_n(x_1, x_2, \ldots, x_n))) \\
&= P(L(w)+1\,,\, P(z\,,\, R(w)))\ .
\end{aligned}$$

(v) Given $w = \sigma(x_1, x_2, \ldots, x_n) = P_{n+1}(n, x_1, x_2, \ldots, x_n)$, in the case $r = 1$,

$$\begin{aligned}
\mathrm{rep}(1,w,y) &= \sigma(y, x_2, x_3, \ldots, x_n) \\
&= P(n\,,\, P_n(y, x_2, x_3, \ldots, x_n)) \\
&= P(n\,,\, P(y, P_{n-1}(x_2, x_3, \ldots, x_n))) \\
&= P(L(w)\,,\, P(y, R^2(w)))
\end{aligned}$$

and in the case $r \ge 2$,

$$\begin{aligned}
\mathrm{rep}(r,w,y) &= \sigma(x_1, \ldots, x_{r-1}, y, x_{r+1}, \ldots, x_n) \\
&= P(n\,,\, P_n(x_1, \ldots, x_{r-1}, y, x_{r+1}, \ldots, x_n)) \\
&= P(n\,,\, P(x_1, P_{n-1}(x_2, \ldots, x_{r-1}, y, x_{r+1}, \ldots, x_n))
\end{aligned}$$

but $\mathrm{del}(w) = (x_2, \ldots, x_n)$ and so

$$\begin{aligned}
\mathrm{rep}(r-1, \mathrm{del}(w), y) &= \sigma(x_2, \ldots, x_{r-1}, y, x_{r+1}, \ldots, x_n) \\
\text{so} \qquad \mathrm{rep}(r,w,y) &= P(L(w)\,,\, P(x_1\,,\, \mathrm{rep}(r-1, \mathrm{del}(w), y)))\ .
\end{aligned}$$

It will be observed that Case (v) is a definition by primitive recursion over r and otherwise every step in all these cases is an operation we already know to be primitive recursive. ■

C A simplified programming language

We consider a programming language, modelled loosely on such languages as Pascal or C. We will strip the language down by removing most of those features of "real" languages which are included only for convenience or efficiency. The only data type, for instance, will be natural numbers. We allow our language to be more powerful than "real" ones in that there are no memory restrictions: it can work with numbers which are arbitrarily large and use as much memory as it pleases.

A program will consist of a finite number of *routines*, one of which is designated the *main* one. Each routine will compute (with the help of the others) some partial function $\mathbb{N}^n \to \mathbb{N}$, and the function deemed to be computed by the program will be that one computed by its main routine.

A routine will have a set $x_1, x_2, \ldots, x_n$ of *variables* and a sequence $S_1, S_2, \ldots, S_k$ of *statements*. Each of the statements modifies the variables in some way, and when the statements have been executed a value is returned, the *value* computed by that routine.

C.1 Definition of the language (Syntax)

A **program** has the following form: one or more routines of which one is designated the main one.

`main routine; routine; ... ; routine .`

There must be at least one routine.

A **routine** has the following form

`name(`$\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_m; \mathtt{x}_{m+1}, \ldots, \mathtt{x}_n$`) {` statement; statement; ... ; return-statement`}`

Here $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_m; \mathtt{x}_{m+1}, \ldots, \mathtt{x}_n$ are the *variables* of which $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_m$ are the *arguments* and $\mathtt{x}_{m+1}, \ldots, \mathtt{x}_n$ the *local variables*. Either or both of these may be empty, that is, $n \geq m \geq 0$. The variable symbols are all distinct. There are zero or more statements. The names of the routines which make up a program must all be different.

A **statement** has one of the following forms: it is an *assignment statement*, a *return statement*, a *conditional statement* or a *while* statement.

An **assignment statement** has one of the forms

`x := 0`
`x := x+1`
`x := y`
`x := y+z`
`x := y`$\dot{-}$`z`
`x := y*z`
`x := F(`$\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_m$`) .`

Here `x`, `y`, `z`, $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_m$ are variable symbols, not necessarily different, `F` is the name of

one of the routines comprising the program and `F` has m arguments. The star (`*`) is used to denote multiplication.

A **return statement** is of the form

`return x` (where x is a variable).

(It will be observed above that it is part of the definition of a routine that it must end with a return statement; this does not stop the routine from having other return statements in other places as well.)

A **conditional statement** is of the form

`if (x≠0) {`$\mathtt{S}_1$`;` $\mathtt{S}_2$`; ... ;` $\mathtt{S}_k$`}` (where x is a variable and $S_1, S_2, \ldots, S_k$ are zero or more statements).

A **while statement** is of the form

`while (x≠0) {`$\mathtt{S}_1$`;` $\mathtt{S}_2$`; ... ;` $\mathtt{S}_k$`}` (where x is a variable and $S_1, S_2, \ldots, S_k$ are zero or more statements).

C.2 What a program does (Semantics)

Suppose the program's main routine has m variables. Then the program computes the function $\mathbb{N}^m \to \mathbb{N}$ defined by its main routine.

The function, f say, $\mathbb{N}^m \to \mathbb{N}$ defined by a routine

`F(`$\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_m; \mathtt{x}_{m+1}, \ldots, \mathtt{x}_n$`) {`$\mathtt{S}_1$`; `$\mathtt{S}_2$`; ... ; `$\mathtt{S}_k$`}`

is specified as follows. To compute `F(`$\xi_1, \xi_2, \ldots, \xi_m$`)`, we set the initial values of its arguments $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_m$ to the given values and initialise the local variables $\mathtt{x}_{m+1}, \ldots, \mathtt{x}_n$ to zero. Then we execute the statements $\mathtt{S}_1$`; `$\mathtt{S}_2$`; ...; `$\mathtt{S}_k$ in order. These may modify the variables, in fact usually do so. As soon as a return statement is encountered, the execution of the routine stops and the value of the variable mentioned in that statement is the returned value.

The assignment statements

`x := 0` `x := x+1` `x := y` `x := y+z` `x := y`$\dot{-}$`z`

`x := y*z` `x := F(`$\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_m$`)`

all do what they say: compute the expression on the right hand side, and set the value of the left hand side variable to this value.

The return statement

`return x`

terminates execution of the routine which contains it and returns the value x.

The conditional statement

`if (x≠0) {`$\mathtt{S}_1$`; `$\mathtt{S}_2$`; ...; `$\mathtt{S}_k$`}`

executes the statements $\mathtt{S}_1, \mathtt{S}_2, \ldots, \mathtt{S}_k$ if `x` is nonzero and ignores them if it is zero.

The while statement

`while (x≠0) {`$\mathtt{S}_1$`; `$\mathtt{S}_2$`; ...; `$\mathtt{S}_k$`}`

repeatedly executes the statements $\mathtt{S}_1;\mathtt{S}_2;\ldots;\mathtt{S}_k$ while `x` is nonzero. The value of `x` is tested before the statements are executed, so if `x` is zero coming in, the statements are not executed at all.

Note The point of all this of course is that a function can be computed by a program in the way described above if and only if it is partial recursive. For the time being, until this is proved, let us call these functions *programmable*.

C.3 Examples

Two ways of computing the factorial function $f(n) = n!$ illustrate the various features of the language.

(1) We compute a value by multiplying together all the numbers from 1 to n. As a first approximation,

```
1  main F(n; v,k) {
2      v := 1;
3      k := 1;
4      while(k<n) {
5          k := k+1;
6          v := v*k;
7      }
8      return v
9  }
```

This need a little fixing up. Firstly, `v := 1` is not one of the allowed statement forms; replace it by `v := 0 ; v:=v+1` and do the same for `k:=1`. Also, `while (k<n)` must be replaced. We define an extra local variable `c` and use `while (c≠0)` where `c:=n`$\dot{-}$`k`. The final version is

```
1  main F(n; v,k,c) {
2      v := 0; v := v+1;
3      k := 0; k := k+1;
4      c := n-.k;
5      while(c≠0) {
6          k := k+1;
7          v := v*k;
8          c := n-.k
```

```
    }
    return v
}
```

(2) For our second method we allow the routine `F` to call itself "recursively" (in the Computer Science sense). We compute f by: if $n = 0$ then $f(n) = 1$, otherwise $f(n) = n * f(n-1)$.

```
main F(n; v,w,x) {
    v := 0; v := v+1;
    if(n≠0) {
        x := n ∸ 1;
        w := F(x);
        v := v*w
    }
    return v
}
```

C.4 Theorem

A function is programmable if and only if it is partially recursive.

Proof. First we prove

(C.4a) *All partial recursive functions are programmable.*

We show this by induction over the way a partial recursive function is constructed in Definition A.2. ▶ A.2

(i) The successor function is implemented thus:

```
main suc(x) {
    x := x+1;
    return x
}
```

(ii) The projection function $(x_1, x_2, \ldots, x_n) \mapsto x_i$ is implemented thus:

```
main p(x1, x2, ..., xn) {
    return xi
}
```

(iii) Addition is implemented:

```
1  main add(x,y; z) {
2      z := x+y;
3      return z
4  }
```

(iv) and **(v)** Natural subtraction and multiplication are implemented in the same way.

(vi) Suppose that $f : \mathbb{N}^n \to \mathbb{N}$ is obtained by substitution, $f = h(g_1, g_2, \ldots, g_m)$ say, where h and $g_1, g_2, \ldots, g_m$ are programmable. Make a new program which consists of all the routines of all the programs for these functions (with name changes as necessary to avoid clashes). Suppose that the routines which were the mains of these programs are named `H`, $\mathtt{G}_1, \mathtt{G}_2, \ldots, \mathtt{G}_m$ respectively. To all this, add one more routine

```
1  main F(x_1, x_2, ..., x_n, y_1, y_2, ..., y_m, z) {
2      y_1 := G_1(x_1, x_2, ..., x_n);
3      y_2 := G_2(x_1, x_2, ..., x_n);
4           ⋮
5      y_m := G_m(x_1, x_2, ..., x_n);
6      z := H(y_1, y_2, ..., y_m);
7      return z
8  }
```

(vii) Finally suppose that $f : \mathbb{N}^n \to \mathbb{N}$ is obtained by minimalisation

$$f(x_1, x_2, \ldots, x_n) = \min_y \{\, g(y, x_1, x_2, \ldots, x_n) = 0 \,\},$$

where g is programmable. Again, start with the program to compute g and call its main routine `G`. Add a new main routine as follows:

```
1  main F(x_1, x_2, ..., x_n, y, c) {
2      y := 0;
3      c := G(y, x_1, x_2, ..., x_n);
4      while (c≠0) {
5          y := y+1;
6          c := G(y, x_1, x_2, ..., x_n);
7      }
8      return y;
9  }
```

And now we prove

(C.4b) *All programmable functions are partial recursive.*

This is quite long and complicated. We will start by making a few minor changes to our program which will make it easier to deal with. First we break up each function assignment statement

`w := F(`$x_1, x_2, \ldots, x_m$`)`

into two simpler statements, a *Function call* and a *Get value*:

`F(`$x_1, x_2, \ldots, x_m$`)`

`w:=value`

We add a dummy `Endwhile` statement to the end of each while group. We also assume that all the statements have been numbered (by numbers ≥ 1) for reference. They do not necessarily have to be numbered consecutively (though this is usually the most convenient thing to do), but every statement must have a different number. The result of performing these preliminaries on our two example programs above are

```
1  main F(n; v,k,c) {
2      v := 0;
3      v := v+1;
4      k := 0;
5      k := k+1;
6      c := n∸k;
7      while(c≠0) {
8          k := k+1;
9          v := vk;
10         c := n∸k;
11         Endwhile;
12     return v }
```

and

```
1  main F(n; v,w,x) {
2      v := 0;
3      v := v+1;
4      if(n≠0) {
5          x := n∸1;
6          F(x);
7          w := value;
8          v := vw }
9      return v }
```

From now on we will be making use of the functions σ, len, ent, del, add and rep defined in Definition B.16 and Lemma B.17. Remember that we know that these are all primitive recursive. ▶ B.16

We can represent the current state of calculation of a particular routine by the current values of all its variables together with the number r of the statement it is up to. For a routine $\texttt{F(x}_1, \texttt{x}_2, \ldots, \texttt{x}_m; \texttt{x}_{m+1}, \ldots, \texttt{x}_n\texttt{)}$ then, we represent its internal state by the number $\boldsymbol{x} = \sigma(r, x_1, x_2, \ldots, x_m)$.

At any stage in a calculation, the main routine may have called another, which in turn may have called another and so on. The state of the whole machine is represented by a sequence $\boldsymbol{s} = \sigma(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_d)$, where $\boldsymbol{x}_d$ is the state of the main routine, working down the hierarchy to $\boldsymbol{x}_1$, the state of the currently executing routine. We will call $\boldsymbol{s}$ the *stack*. We will call its first entry $\boldsymbol{x}_1$ the *top of the stack* and denote it $\mathrm{tos}(\boldsymbol{s})$ (in fact $\mathrm{tos}(\boldsymbol{s}) = \mathrm{ent}(1, \boldsymbol{s})$, so tos is primitive recursive).

We will refer to entries on the top of the stack frequently, so an abbreviation is useful:

$$\mathrm{tent}(i, \boldsymbol{s}) \;=\; \mathrm{ent}(i+1, \mathrm{tos}(\boldsymbol{s}))\,.$$

(the offset in the index i here is so that the ith variable x_i on the top of the stack is $\mathrm{tent}(i, \boldsymbol{s})$ and the current statement number r is $\mathrm{tent}(0, \boldsymbol{s})$). It will also be convenient to define $\mathrm{trep}(i, \boldsymbol{s}, y)$ to be the result of replacing the $(i+1)$st entry on the top of the stack by y:

$$\mathrm{trep}(i, \boldsymbol{s}, y) \;=\; \mathrm{rep}(1, \boldsymbol{s}, \mathrm{rep}(i+1, \mathrm{tos}(\boldsymbol{s}), y))\,.$$

It is clear that all these functions are primitive recursive.

We now define a *next state function* $\mathrm{Next} : \mathbb{N}^n \to \mathbb{N}$. We are going to define it so that, if $\boldsymbol{s}$ represents a stack (which, remember, specifies the statement about to be executed amongst other things), then $\mathrm{Next}(\boldsymbol{s})$ represents the stack after that one statement has been executed. We will also show that Next is primitive recursive.

The action of a routine is as follows. It expects to find its variables set up on top of the stack:

$$\text{top of stack} = (r, x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_m)$$

where

r is the number of the first statement of this routine,

$x_1, x_2, \ldots, x_n$ are the values of its arguments and

$x_{n+1}, \ldots, x_m$, the local variables, are zero at this time.

Upon completion, it removes its variables from the stack and leaves the pair (r', v) there, where v is the return value and r' is the number of the get-value statement it is to return to. (If it is returning from the initial invocation of the main routine, there is nowhere to return to, and we set $r' = 0$.)

$\mathrm{Next}(\boldsymbol{s})$ is computed as follows.

Get the number of the statement to be executed, $r = \mathrm{tent}(0, \boldsymbol{s})$

then execute the function, $\bar{S}_r$ say, corresponding to statement number r.

Assuming the statements are numbered 1,2,...,k, we have

$$\mathrm{Next}(\boldsymbol{s}) = \mathrm{eq}(r,0).\boldsymbol{s} + \mathrm{eq}(r,1).\bar{S}_1(\boldsymbol{s}) + \mathrm{eq}(r,2).\bar{S}_2(\boldsymbol{s}) + \cdots + \mathrm{eq}(r,k).\bar{S}_k(\boldsymbol{s}).$$

The first term in this sum, $\mathrm{eq}(r,0).\boldsymbol{s}$, is so that, when the program terminates by exiting from the main routine, we have $r = 0$ and the state $\boldsymbol{s}$ remains unchanged from then on.

It remains to define the function $\bar{S}$ for each kind of statement and show that these functions are primitive recursive.

Suppose first that S is the assignment statement $\mathtt{x}_i\mathtt{=x}_j\mathtt{+x}_k$. Then

$$\bar{S}(\boldsymbol{s}) = \mathrm{trep}(0, \boldsymbol{s}', \mathrm{tent}(0,\boldsymbol{s}) + 1)$$

where

$$\boldsymbol{s}' = \mathrm{trep}(i, \boldsymbol{s}, \mathrm{tent}(j, \mathrm{tos}(\boldsymbol{s})) + \mathrm{tent}(k, \mathrm{tos}(\boldsymbol{s}))) \,.$$

Assignment statements of the forms

$$\mathtt{x}_i\mathtt{:=0} \qquad \mathtt{x}_i\mathtt{:=x}_i\mathtt{+1} \qquad \mathtt{x}_i\mathtt{:=x}_j \qquad \mathtt{x}_i\mathtt{:=x}_j \dot{-} \mathtt{x}_k \qquad \mathtt{x}_i\mathtt{:=x}_j\mathtt{x}_k$$

can be implemented in a similar manner. In all cases, $\bar{S}$ is primitive recursive.

Suppose that S is a function call statement, $\mathtt{F(x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_m\mathtt{)}$, where the function being called has variables $\mathtt{y}_1, \mathtt{y}_2, \ldots, \mathtt{y}_m; \mathtt{y}_{m+1}, \ldots, \mathtt{y}_n$ and the number of its first statement is r'. Then the actions to be taken are

- Update the current statement number r, so that the routine will return to the next Get Value statement, $\boldsymbol{s}' = \mathrm{trep}(0, \boldsymbol{s}, \mathrm{tent}(0,\boldsymbol{s}) + 1)$.
- Set up the variables to go on top of the stack (call this $\boldsymbol{t}$), so that
  $$\boldsymbol{t} = \sigma(r', \mathrm{tent}(j_1, (\boldsymbol{s}), \mathrm{tent}(j+2, \boldsymbol{s}), \ldots, \mathrm{tent}(j_n, \boldsymbol{s}), 0, 0, \ldots, 0)$$
- Push these variables on top of the stack: $\boldsymbol{s} \mapsto \mathrm{add}(\boldsymbol{t}, \boldsymbol{s}')$.

Therefore $\bar{S}$ is defined by

$$\bar{S}(\boldsymbol{s}) \;=\; \mathrm{add}(\boldsymbol{t}, \boldsymbol{s}') \,,$$

where $\boldsymbol{t}$ and $\boldsymbol{s}'$ are as above.

A Return statement ($\mathtt{return\ x}_i$) will replace the top of stack $\boldsymbol{x}$ with the pair $\boldsymbol{x}' = (r, x_i)$, where r is the statement number of the Get Value statement to be returned to. This is currently stored in the second $\boldsymbol{x}$ from the top of the stack. The actions to be taken are

- Get r and x_i, $\quad r = \mathrm{ent}(1, \mathrm{ent}(2, \boldsymbol{s})) \quad , \quad x_i = \mathrm{tent}(i, \boldsymbol{s})$,
- Replace the $\boldsymbol{x}$ on top of the stack by this pair, $\quad \sigma_2(r, x_i)$.

So

$$\bar{S}(\boldsymbol{s}) = \rho_1(\boldsymbol{s}, \sigma_2(r, x_i))$$

where r and x_i are as above (recall that σ_2, the restriction of σ to $\mathbb{N}^2$, is primitive recursive).

(Note: when returning from the main routine, there is only one $\boldsymbol{x}$ on the stack, so ent$(2, \boldsymbol{s})$ and therefore r will be zero. This will be used to determine when the entire calculation is finished).

A get value statement `x`$_i$`=value` always occurs immediately following a function call and expects to find $\boldsymbol{x} = (r, v)$ on top of the stack, where r is its own statement number, and v is the return value of the function just computed. The actions to be taken are

- Get v, $\quad v = \text{tent}(2, \boldsymbol{s})$
- Remove the top of stack, $\quad \boldsymbol{s}' = \text{del}(\boldsymbol{s})$
- Change the value of `x`$_i$ to v, $\quad$ new $\boldsymbol{s} = \text{trep}(i, \boldsymbol{s}', v))$

So we have

$$\bar{S}(\boldsymbol{s}) = \text{trep}(i, \boldsymbol{s}', v))\,,$$

where $\boldsymbol{s}'$ and v are as above.

Let us look at an `if` statement in context

`if (x`$_i \neq$`0) {S`$_1$ `; S`$_2$ `; ... ; S`$_k$ `} ;`
`S`$_{k+1}$

This should look at the variable x_i. If it is nonzero, the statement number on top of the stack should be changed to the number of `S`$_i$, otherwise it should be changed to the number of `S`$_{k+1}$. Calling these statement numbers r_1 and r_2 respectively, then, first, compute a number c which is 1 if $x_i \neq 0$ and 0 otherwise,

$$c = 1 \dot{-} \text{tent}(i, \boldsymbol{s})$$

then set

$$\bar{S}(\boldsymbol{s}) = c\ \text{trep}(0, \boldsymbol{s}, r_1) + (1 \dot{-} c)\,\text{trep}(0, \boldsymbol{s}, r_1)\,.$$

(Some nitpicking: Because every routine must end with a `return` statement, the statement `S`$_{k+1}$ must actually exist. It is possible, if unlikely, that there are no statements between the braces (i.e. $k = 0$). In this case r_1 and r_2 are both the statement number of S_{k+1}.)

A *While* statement is implemented in exactly the same fashion — it is the presence of the *Endwhile* statement which makes the difference between the two types of statement.

An *Endwhile* statement simply makes sure that the next statement to be executed is the test at the head of the statement. If this has statement number r, then the Endwhile statement is

$$\bar{S}(\boldsymbol{s}) = \text{trep}(0, \boldsymbol{s}, r)\,.$$

This completes the rather complicated definition of the Next function as a primitive recursive function.

Remembering that we are supposing that the program is designed to compute a function $f : \mathbb{N}^n \to \mathbb{N}$ and has a main routine of the form

$$\texttt{F(x}_1, \texttt{x}_2, \ldots, \texttt{x}_m; \texttt{x}_{m+1}, \ldots, \texttt{x}_n\texttt{)},$$

we define a function Step : $\mathbb{N}^{n+1} \to \mathbb{N}$ so that $\mathrm{Step}(t, x_1, x_2, \ldots, x_m)$ is the state $\boldsymbol{s}$ of the calculation after t statements have been executed, when the function was called with the values $x_1, x_2, \ldots, x_m$ to start off. This is now easy: the function is initialised ($t = 0$) to the initial state of the stack (ready to set the main routine going), and then defined by primitive recursion using the Next function.

$$\begin{aligned} \mathrm{Step}(0, x_1, x_2, \ldots, x_n) &= \sigma_1(\boldsymbol{x}) \qquad \text{where} \quad \boldsymbol{x} = \sigma_{m+1}(r, x_1, x_2, \ldots, x_n, 0, 0, \ldots, 0)\,, \\ \mathrm{Step}(t+1, x_1, x_2, \ldots, x_n) &= \mathrm{Next}(\mathrm{Step}(t, x_1, x_2, \ldots, x_n))\,. \end{aligned}$$

We can determine from this when the entire calulation is ended and what the value is. At the end of the calculation (and not before) the machine has just executed a return statement from the main routine. At this stage, and not before, the next statement number r on top of the stack is 0. Also at this stage, the top of the stack (actually the only thing on the stack) is the pair (r, v), where v is the value computed. Thus, if T is the step to terminate at, we have

$$\begin{aligned} f(x) &= \mathrm{tent}(1, \mathrm{Step}(T, x_1, x_2, \ldots, x_m)) \\ &\qquad \text{where} \quad T = \min_t\{\mathrm{tent}(0, \mathrm{Step}(t, x_1, x_2, \ldots, x_m))) = 0\}\,. \end{aligned}$$

Since the Step function is primitive recursive, this shows that f is partial recursive. ■

We have also observed that the functions tent and Step are primitive recursive. Thus we have the remarkable result:

C.5 Corollary

Any partial recursive function $f : \mathbb{N}^n \to \mathbb{N}$ can be written in the form

$$f(\boldsymbol{x}) = p(\min_t\{q(t, \boldsymbol{x}) = 0\}, \boldsymbol{x}),$$

where p and q are primitive recursive functions $\mathbb{N}^{n+1} \to \mathbb{N}$.

C.6 Corollary

A function $\mathbb{N}^n \to \mathbb{N}$ is recursive if and only if it can be obtained by a finite number of applications of the operations of

- substitution, and
- minimalisation applied to a regular function

from the list:—

(1) $x \mapsto x+1$, the successor function $s : \mathbb{N} \to \mathbb{N}$

(2) $(x_1, x_2, \ldots, x_n) \mapsto x_i$, the projection function $\pi_{n,i} : \mathbb{N}^n \to \mathbb{N}$

(3) $(x, y) \mapsto x + y$, addition $\mathbb{N}^2 \to \mathbb{N}$

(4) $(x, y) \mapsto x \dot{-} y$, natural subtraction $\mathbb{N}^2 \to \mathbb{N}$

(5) $(x, y) \mapsto xy$, multiplication $\mathbb{N}^2 \to \mathbb{N}$

C.7 Some extensions

The language is flexible enough for us to write most algorithms we will need with some ease. There are some minor extensions which increase its convenience. It will be seen that each of these extensions can be incorporated into the language because whatever can be done with them can also be done without.

(i) *More complicated assignment statements.* We can include assignments statements such as

```
x := x + y*z*F(x+1,x*y+G(z))
```

since they can easily be built up from a sequence of allowed assignment statements, at the cost perhaps of introducing a few more local variables.

(ii) *We may include any function already known to be partial recursive.* For example

`x :=` $\lfloor\sqrt{\texttt{y}}\rfloor$

Simply write down the program to compute that function, then add all its routines to the program being written (changing routine names and statement numbers if necessary).

(iii) Any recursive predicate can be included in the test part of an *If* or *While* statement. For example

```
while(x<y or x+z≥y) {...
```

Simply make a new local variable to represent the test and compute it first as the boolean function of that test:

```
c := ((x+1)∸y).(y∸(x+z)) ;

while (c≠0) {...
```

The computation of `c` will have to be repeated at the very end of the while statement.

(iv) *If statements may have "else" sections if you like.* For example

```
if (c≠0) {S1; S2;...; Sk} ;

else {Sk+1; Sk+2;...; Sl} ;
```

is implemented as

`if (c≠0) {`$\mathtt{S}_1$`; `$\mathtt{S}_2$`;...; `$\mathtt{S}_k$`} ;`

`if (c=0) {`$\mathtt{S}_{k+1}$`; `$\mathtt{S}_{k+2}$`;...; `$\mathtt{S}_l$`} ;`

(v) *Computations may be included in Return statements.* For example

```
return x+y;
```

is implemented as

```
v:=x+y;
return v;
```

C.8 Programming primitive recursive functions

The programming language we have been discussing makes it very easy to specify recursive functions. It would be nice if we could specify primitive recursive functions in a similar fashion. We now show that, by making a couple of small changes to our idealised programming language, it becomes a language for specifying primitive recursive functions.

The changes we will make are:

(1) Recursive routine calls are not allowed — I am here using "recursive" in the computer-science sense: routines may not call themselves, nor may they call other functions which call themselves and so on. We will ensure this by specifying that our routines must be numbered from 1 (for the main routine) down to K say (for the "deepest" one) and making the rule that routine F_i may only call routine F_j if $j > i$.

(2) No *while* statements are allowed. Instead we have the simple `for` statement

`for(x,n) {`$\mathtt{S}_1$`; `$\mathtt{S}_2$`; ...; `$\mathtt{S}_k$` }` .

Execution of this statement causes the statements $\mathtt{S}_1, \mathtt{S}_2, \ldots, \mathtt{S}_k$ to be executed `n` times, for values of `x` from 1 to `n`. The statements $\mathtt{S}_1, \mathtt{S}_2, \ldots, \mathtt{S}_k$ are allowed to use the values of `x` and `n` but not to change them. If $\mathtt{n} = 0$, these statements are not executed at all.

(3) As before, the last statement of a routine must be a `return` statement, but now that is the only place a `return` statement is allowed to be. In particular, `return` statements may not appear inside `if` or `for` statements. (This requirement is not strictly necessary; the next theorem would still be true without it. However it does no harm, and it makes the theorem easier to prove.)

Let us call our programming language, so modified, the *primitive programming language*.

C.9 Theorem

A function may be computed by the primitive programming language above if and only if it is primitive recursive.

Proof. To see that any primitive recursive function can be computed by the primitive programming language, we use the same proof as in Theorem C.4, and induction over the ▶ C.4

construction of the function as in Definition B.3. The methods given for computing the basic functions (the successor, zero and projection functions) and functions defined by substitution are valid in our primitive language also, so it remains to show that a definition by primitive recursion can be so programmed. ► B.3

So suppose the function $f : \mathbb{N}^n \to \mathbb{N}$ is defined by

$$\begin{aligned} f(0, \boldsymbol{y}) &= g(\boldsymbol{y}) \\ f(x+1, \boldsymbol{y}) &= h(x, f(x, \boldsymbol{y}), \boldsymbol{y}) \,, \end{aligned}$$

where g and h are primitive recursive. We may suppose that g and h are programmable by routines `G` and `H` in our primitive language. Then f may be programmed by

```
1  main F(x,y_1,y_2,...,y_n;v,i) {
2      v := G(y_1,y_2,...,y_n);
3      for (i,x) {v:=H(x,v,y_1,y_2,...,y_n)};
4      return v
5  }
```

Now we come to the proof that any function which is primitive programmable is primitive recursive. The proof here is more straightforward than is the case for Theorem C.4. We prove that, for any routine `F` in the program, the returned value is a primitive recursive function of its arguments. Remembering that our routines are numbered $\mathtt{F}_1, \mathtt{F}_2, \ldots, \mathtt{F}_K$ so that, whenever a statement of routine $\mathtt{F}_i$ calls $\mathtt{F}_j$ we must have $i < j$, we may work by induction back from the "deepest" routine $\mathtt{F}_K$ to the main one $\mathtt{F}_1$ and assume, when we are proving the result about any particular routine, that it is true for any deeper routines it may call. ► C.4

So let us consider a particular routine

$$\mathtt{F(x}_1\mathtt{,x}_2\mathtt{,}\ldots\mathtt{,x}_m\mathtt{;x}_{m+1}\mathtt{,x}_{m+2}\mathtt{,}\ldots\mathtt{,x}_n\mathtt{)\,\{S}_1\mathtt{;S}_2\mathtt{;}\ldots\mathtt{;S}_k\mathtt{\}}\,.$$

Let us write $\boldsymbol{x}$ for the string of variables' values $(x_1, x_2, \ldots, x_n)$ at any time (note that that includes the values of the local variables). For each statement $\mathtt{S}_i$, let us write $s_i : \mathbb{N}^m \to \mathbb{N}^m$ for the effect that statement has on the variables; that is, if the variables are $\boldsymbol{x}$ just before executing the statement, then they are $s_i(\boldsymbol{x})$ after it.

Then the action of all the statements but the last has the combined result $s_{k-1} \circ s_{k-2} \circ \cdots \circ s_1(\boldsymbol{x})$, where here $\boldsymbol{x}$ is the initial state of the variables, $(x_1, x_2, \ldots, x_n, 0, 0, \ldots, 0)$.

The last statement is the `return` statement. Supposing it is $\mathtt{return\ x}_j$, then this is simply the projection function $\pi_{m,j}$.

Writing ext for the "setting up" function, which extends $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ to $(x_1, x_2, \ldots, x_n, 0, 0, \ldots, 0)$, we see that the function computed by `F` is

$$\pi_{m,j} \circ s_{k-1} \circ s_{k-2} \circ \cdots \circ s_1 \circ \mathrm{ext} \ .$$

Now ext and $\pi_{m,j}$ are primitive recursive, so it remains to show that the action s of each of the individual statements `S` is primitive recursive.

If `S` is one of the assignment statements this is obvious.

If `S` is an `if` statement,

$$\texttt{if(x}_i \neq \texttt{0) \{S}_1\texttt{; S}_2\texttt{; ...; S}_k\texttt{\}}$$

then, as before, the result of executing $\texttt{S}_1, \texttt{S}_2, \ldots, \texttt{S}_k$ (if they are executed) is to convert $\boldsymbol{x}$ to $s_k \circ s_{k-1} \circ \cdots \circ s_1(\boldsymbol{x})$. Therefore the effect of the entire `if` statement is

$$\boldsymbol{x} \;\mapsto\; (1 \dot{-} c)(s_k \circ s_{k-1} \circ \cdots \circ s_1(\boldsymbol{x})) \;+\; c\,\boldsymbol{x} \quad \text{where } c = \mathrm{eq}(x_i, 0)\,.$$

If `S` is a `for` statement,

$$\texttt{for(x}_i\texttt{,x}_j\texttt{) \{S}_1\texttt{; S}_2\texttt{;...; S}_k\texttt{\}}$$

we enter the loop with $\boldsymbol{x}$ initialised by setting $x_i = 0$. Then a single pass through the loop consists of replacing x_i with $x_i + 1$ and then replacing that $\boldsymbol{x}$ with $s_k \circ s_{k-1} \circ \cdots \circ s_1(\boldsymbol{x})$. Thus, defining $\varphi(r, \boldsymbol{x}) : \mathbb{N}^{m+1} \to \mathbb{N}^m$ to be the string of variables after r iterations of the loop, we have

$$\begin{aligned} \varphi(0, \boldsymbol{x}) &= \boldsymbol{x} \text{ with } x_i \text{ changed to } 0\,, \\ \varphi(r+1, \boldsymbol{x}) &= s_k \circ s_{k-1} \circ \cdots \circ s_1(\varphi(r, \boldsymbol{x}')) \quad \text{where } \boldsymbol{x}' \text{ is } \boldsymbol{x} \text{ with } x_i \text{ incremented.} \end{aligned}$$

This is obviously primitive recursive, and the function s for the entire For statement is just $\varphi(x_j, \boldsymbol{x})$.

Finally, if `S` is a function call, $\texttt{x}_i$ `= F(`$\texttt{x}_{j_1}$`,` $\texttt{x}_{j_2}$`,...,` $\texttt{x}_{j_p}$`)` say, where `F` is defined

$$\texttt{F(y}_1\texttt{,y}_2\texttt{,...,y}_p\texttt{;y}_{p+1}\texttt{,y}_{p+2}\texttt{,...,y}_q\texttt{) \{S}_1\texttt{; S}_2\texttt{; ...; S}_k\texttt{\}}$$

most of the work has been done for us already: we have proved that the function, f say, which takes the values $(y_1, y_2, \ldots, y_p)$ of the arguments to the output value, v say, is primitive recursive. The effect the function call has on the variables $\boldsymbol{x}$ is to first produce $(y_1, y_2, \ldots, y_p) = (x_{j_1}, x_{j_2}, \ldots, x_{j_p})$, followed by computing $v = f(y_1, y_2, \ldots, y_p)$ and finally replacing x_i in $\boldsymbol{x}$ by v. All these operations are primitive recursive. ■

D The Ackermann function

D.1 Remarks

Are there any recursive functions which are not primitive recursive? In this section we answer this question with "yes" by constructing the Ackermann function as a recursive function and proving that it is not primitive recursive.

Consider the definitions of addition, multiplication and exponentiation:

$$\begin{aligned} 0+y &= y & (x+1)+y &= \mathrm{suc}(x+y) \\ 0y &= 0 & (x+1)y &= xy+y \\ y^0 &= 1 & y^{x+1} &= y^x.y \end{aligned}$$

We could continue this sequence by defining a few new functions,

$$\begin{aligned} f(0,y) &= 1 & f(x+1,y) &= y^{f(x,y)} \\ g(0,y) &= 1 & g(x+1,y) &= f(g(x,y),y) \end{aligned}$$

and so on.

So let us define a sequence a_m of functions $\mathbb{N}^2 \to \mathbb{N}$ inductively, thus:

$$\begin{aligned} a_0(x,y) &= x+y \\ a_1(0,y) &= 0 & a_1(x+1,y) &= a_0(a_1(x,y),y) \\ a_{m+1}(0,y) &= 1 & a_{m+1}(x+1,y) &= a_m(a_{m+1}(x,y),y) \qquad \text{for all } m \geq 1. \end{aligned}$$

Then a_0 is addition, a_1 is multiplication and a_2 exponentiation. We see that all these functions are primitive recursive and so recursive. Now let us put the subscript in as another argument: define a function $F : \mathbb{N}^3 \to \mathbb{N}$ by $F(m,x,y) = a_m(x,y)$. We can provide a cleaner looking "recursive" definition of this:

$$\begin{aligned} F(0,x,y) &= x+y \\ F(1,0,y) &= 0 \\ F(m,0,y) &= 1 && \text{for all } m \geq 2 \\ F(m+1,x+1,y) &= F(m,F(m+1,x,y),y) && \text{for all } m \text{ and } x. \end{aligned}$$

This function is recursive but not primitive recursive (recursive is obvious, not primitive recursive is not).

We will prove these facts here for the Ackermann function, which is a closely related (and simpler) function A given by ...

D.2 Definition

In this section, A will be the function $\mathbb{N}^2 \to \mathbb{N}$ given by

$$\begin{aligned} A(0,x) &= x+2 \\ A(m+1,0) &= A(m,A(m,0)) \\ A(m+1,x+1) &= A(m,A(m+1,x))\,. \end{aligned}$$

D.3 Theorem

There is a function $\mathbb{N} \to \mathbb{N}$ which is recursive but not primitive recursive, namely $x \mapsto A(x,x)$.

Proof. Firstly, it is easily seen that A is recursive by writing a program for it:

```
1  main A(m,x) {
2      if (m=0) {return x+2 };
3      if (m>0 and x=0) {return A(m-.1,A(m-.1,0))}
4      if (m>0 and x>0) {return A(m-.1,A(m,x-.1))}
5      return 0
6  }
```

(The program can never reach that `return 0` statement at the end; it is just there to obey the rules.)

It follows that $x \mapsto A(x,x)$ is recursive also.

The proof that this function is not primitive recursive is in several steps.

Step 1

(i) $A(1,x) = 2x+4$ for all $x \in \mathbb{N}$.

(ii) $A(m,x) \geq m+x+2$ for all $m,x \in \mathbb{N}$.

(iii) $A(m,x) > x$ for all $m,x \in \mathbb{N}$.

Proof (i) follows directly from the definition above by induction over x.

(ii) is proved by induction over the pair $\langle m,x\rangle$.

$$\begin{aligned} A(0,x) &= x+2 \;\geq\; 0+x+2 \\ A(m+1,0) &= A(m,A(m,0)) \;\geq\; m+A(m,0)+2 \\ &\geq m+(m+0+2)+2 \;\geq\; (m+1)+2\,. \\ A(m+1,x+1) &= A(m,A(m+1,x)) \;\geq\; m+A(m+1,x)+2 \\ &\geq m+(m+1+x+2)+2 \;\geq\; (m+1)+(x+1)+2,. \end{aligned}$$

(iii) follows immediately from (ii).

Step 2

The function A is strictly monotonic increasing in both variables.

Proof (i) First we show that $x \mapsto A(m,x)$ is monotonic increasing, by induction over m. Zerothly, $A(0,x) = x+2$, so the result is obvious in the case $m = 0$. Now suppose that $x \mapsto A(m,x)$ is monotonic increasing; we prove that $x \mapsto A(m+1,x)$ is also. For all x,

$$\begin{aligned} A(m+1,x+1) &= A(m,A(m+1,x)) && \text{by definition of } A \\ &> A(m+1,x) && \text{by Step 1(iii)} \end{aligned}$$

(ii) To show that $m \mapsto A(m,x)$ is monotonic increasing, we show that $A(m,x) < A(m+1,x)$ for all m and x. Now

$$\begin{aligned} A(m+1,0) &= A(m,A(m,0)) \\ &> A(m,0) && \text{by Step1(iii)} \\ A(m+1,x+1) &= A(m,A(m+1,x)) \\ &> A(m,x) && \text{by Part (i), since } A(m+1,x) > x \end{aligned}$$

Step 3

$$A(m+1,x) \geq A(m,x+1) \qquad \text{for all } m \text{ and } x.$$

Proof

$$\begin{aligned} A(m+1,0) &= A(m,A(m,0)) \\ &\geq A(m,1) && \text{since } A(m,0) \geq 1 \\ A(m+1,x+1) &= A(m,A(m+1,x)) \\ &> A(m,x+2) && \text{by Step 1(ii) and Step 2} \end{aligned}$$

Step 4

$$A(m,2x) \leq A(m+1,x) \qquad \text{for all } m \text{ and } x \text{ .}$$

Proof For the case $x = 0$, we have $A(m,0) \leq A(m+1,0)$ by monotonicity. Otherwise,

$$\begin{aligned} A(m+1,x+1) &= A(m,A(m+1,x)) && \text{by definition of } A \\ &\geq A(m,A(1,x)) && \text{by monotonicity} \\ &= A(m,2x+4) && \text{by Step 1(i)} \\ &\geq A(m,2(x+1)) \end{aligned}$$

Step 5

$$A(m+k,x) \geq A(m,kx) \qquad \text{for all } m \text{ and } x.$$

Proof This is overkill. From Step 4 we have

$$A(m+k,x) \;\geq\; A(m,2^k x)$$

and the result follows.

Step 6 (The main step)

Let f be a primitive recursive function $\mathbb{N}^n \to \mathbb{N}$. Then there is a natural number m such that

$$f(\boldsymbol{x}) \;\leq\; A(m,|\boldsymbol{x}|) \qquad\qquad \text{for all } \boldsymbol{x} \in \mathbb{N}^n$$

(Here $|\boldsymbol{x}|$ is just a convenient abbreviation for $x_1 + x_2 + \cdots + x_n$. In the special case where f is a function $\mathbb{N}^0 \to \mathbb{N}$, so that $\boldsymbol{x}$ is empty, we set $|\boldsymbol{x}| = 0$.)

Proof is by induction over the way f is constructed, using Definition B.3. ▶ B.3

If f is any one of the three basic functions listed (the successor function, the zero constant or any of the projection functions), we have $f(\boldsymbol{x}) \leq |\boldsymbol{x}| + 2 = A(0,|\boldsymbol{x}|)$ for all $\boldsymbol{x}$.

Now suppose that f is obtained by substitution, $f = h(g_1, g_2, \ldots, g_k)$. Then there are natural numbers $\mu, m_1, m_2, \ldots, m_k$ such that

$$\begin{aligned} h(\boldsymbol{y}) &\leq A(\mu,|\boldsymbol{y}|) && \text{for all } \boldsymbol{y} \in \mathbb{N}^k \\ g_i(\boldsymbol{x}) &\leq A(m_i,|\boldsymbol{x}|) && \text{for all } \boldsymbol{x} \in \mathbb{N}^n \text{ and } i = 1,2,\ldots,k\ . \end{aligned}$$

Now set $M = \max\{\mu + k + 1, m_1 + 1, m_2 + 1, \ldots, m_k + 1\}$. Then $\mu \leq M - k - 1$ and each $m_i \leq M - 1$, so

$$\begin{aligned} h(\boldsymbol{y}) &\;\leq\; A(M-k-1,|\boldsymbol{y}|) \qquad \text{for all } \boldsymbol{y} \in \mathbb{N}^m, \text{ and} \\ g_i(\boldsymbol{x}) &\;\leq\; A(M-1,|\boldsymbol{x}|) \qquad \text{for all } \boldsymbol{x} \in \mathbb{N}^n \text{ and } i = 1,2,\ldots,k. \end{aligned}$$

Then $$|(g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_k(\boldsymbol{x}))| = g_1(\boldsymbol{x}) + g_2(\boldsymbol{x}) + \cdots + g_k(\boldsymbol{x}) \leq kA(M-1,|\boldsymbol{x}|)$$

So

$$\begin{aligned} f(\boldsymbol{x}) &= h(g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_k(\boldsymbol{x})) \\ &\leq A(M-k-1, |(g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_k(\boldsymbol{x}))|) \\ &\leq A(M-k-1, kA(M-1,|\boldsymbol{x}|)) \\ &\leq A(M-1, A(M-1,|\boldsymbol{x}|)) && \text{by Step 5} \\ &\leq A(M-1, A(M,|\boldsymbol{x}|-1)) && \text{by Step 3} \\ &= A(M,|\boldsymbol{x}|) && \text{by definition of } A. \end{aligned}$$

as required.

(The last two lines in that proof are valid provided $|\boldsymbol{x}| \geq 1$. In the case $|\boldsymbol{x}| = 0$, argue thus: then the third last line is $A(M-1, A(M-1,0))$, and this is equal to $A(M,0)$ by the definition of A.)

Finally, suppose that f is obtained by primitive recursion,

$$f(0, \boldsymbol{y}) \;=\; g(\boldsymbol{y}) \quad \text{and} \quad f(x+1, \boldsymbol{y}) \;=\; h(x, f(x, \boldsymbol{y}), \boldsymbol{y}). \tag{–1}$$

Then we have numbers μ and m such that

$$g(\boldsymbol{y}) \;\leq\; A(\mu, |\boldsymbol{y}|) \qquad \text{for all } \boldsymbol{y} \tag{–2}$$

$$h(x, u, \boldsymbol{y}) \;\leq\; A(m, x + u + |\boldsymbol{y}|\}) \qquad \text{for all } x,\ u \text{ and } \boldsymbol{y}. \tag{–3}$$

Put $M = \max\{\mu, m+2\}$. We show, by induction over x, that $f(x, \boldsymbol{y}) \leq A(M, x + |\boldsymbol{y}|)$. Firstly,

$$f(0, \boldsymbol{y}) = g(\boldsymbol{y}) \;\leq\; A(\mu, |\boldsymbol{y}|) \;\leq\; A(M, 0 + |\boldsymbol{y}|).$$

Now suppose that $f(x, \boldsymbol{y}) \leq A(m, x + |\boldsymbol{y}|)$. Then

$$\begin{aligned}
f(x+1, \boldsymbol{y}) \;&=\; h(x, f(x, \boldsymbol{y}), \boldsymbol{y}) && \text{by (–1)} \\
&\leq\; A(m, x + f(x, \boldsymbol{y}) + |\boldsymbol{y}|) && \text{by (–3)} \\
&\leq\; A(m, x + |\boldsymbol{y}| + A(M, x + |\boldsymbol{y}|)) && \text{by the inductive hypothesis} \\
&\leq\; A(m, 2A(M, x + |\boldsymbol{y}|)) && \text{since } x + |\boldsymbol{y}| \leq A(M, x + |\boldsymbol{y}|) \\
&\leq\; A(m+1, A(M, x + |\boldsymbol{y}|)) && \text{by Step 5} \\
&\leq\; A(M-1, A(M, x + |\boldsymbol{y}|)) && \text{since } M \geq m + 2) \\
&=\; A(M, x + 1 + |\boldsymbol{y}|) && \text{by definition of } A)
\end{aligned}$$

as required. ■

Proof of the theorem. We have already seen that the function $x \mapsto A(x, x)$ is recursive.

Suppose the function were primitive recursive. Then, by the last lemma, there would be a natural number m such that

$$A(x, x) \leq A(m, x) \qquad \text{for all } x \in \mathbb{N}.$$

But choosing $x > m$ gives $A(x, x) > A(m, x)$, a contradiction. ■

D.4 Corollary

The Ackermann function A is not primitive recursive.

E Two fundamental theorems

We start with a lemma which defines partial recursive functions $\mathbb{N} \to \mathbb{N}$ without reference to functions $\mathbb{N}^n \to \mathbb{N}$ for other n.

E.1 Lemma

The set of partial recursive functions $\mathbb{N} \to \mathbb{N}$ is the closure of the following functions

(i) $x \mapsto x + 1$

(ii) L

(iii) R

(iv) $x \mapsto L(x) + R(x)$

(v) $x \mapsto L(x) \dot{-} R(x)$

(vi) $x \mapsto L(x)R(x)$

under the following operations

(a) Composition

(b) From g and h obtain f given by $f(x) = P(g(x), h(x))$

(c) From g obtain f given by $f(x) = \min_y\{g(P(y, x)) = 0\}$

Proof. Let us temporarily refer to the set of functions defined above as **P** and show that this set is in fact the set of partial recursive functions. It is obvious that all such functions are partial recursive, so it is only the converse that we need to prove.

We will use the inverse functions $P_n : \mathbb{N}^n \to \mathbb{N}$ and $\boldsymbol{E}_n : \mathbb{N} \to \mathbb{N}_n$ defined in Section B.13. ▶ B.13
We will also use two results, both easily proved by induction:

Firstly if $g_1, g_2, \ldots, g_n$ are all members of **P** $(n \geq 1)$, then so is $P_n(g_1, g_2, \ldots, g_n)$.

Secondly, for $n \geq 1$, $E_{n+1}(P(y, w)) = (y, E_n(w))$

Now we prove the lemma by proving the slightly stronger result that, if f is any partially recursive function $\mathbb{N}^n \to \mathbb{N}$ (for any n), then $f \circ \boldsymbol{E}_n : \mathbb{N} \to \mathbb{N}$ is a member of **P**. This contains the required result, since $\boldsymbol{E}_1$ is the identity. We prove this by induction over the construction of f as given in Definition A.2. ▶ A.2

(1) If f is the successor function, then it is a member of **P** by definition.

(2) If f is the projection function $\pi_{n,i} : \mathbb{N}^n \to \mathbb{N}$, that is, $(x_1, x_2, \ldots, x_n) \mapsto x_i$, then (from the definition of $\boldsymbol{E}_n$),

$$f\boldsymbol{E}_n = LR^{i-1} \quad \text{if } i < n \qquad \text{and} \qquad f\boldsymbol{E}_n = R^n \quad \text{if } i = n\ .$$

In either case $f\boldsymbol{E}_n$ is a member of **P**.

(3) If f is addition, $f(x,y) = x + y$, then $f\boldsymbol{E}_2(w) = L(w) + R(w)$ and this is a member of **P** by its definition.

(4) and (5) The same argument holds if f is natural subtraction or multiplication.

(6) Now suppose that f is obtained by substitution, $f = h(g_1, g_2, \ldots, g_m) : \mathbb{N}^n \to \mathbb{N}$ say, where $h\boldsymbol{E}_m$ and $g_1\boldsymbol{E}_n$, $g_2\boldsymbol{E}_n$, ... , $g_m\boldsymbol{E}_n$ are all members of **P**. Then

$$f\boldsymbol{E}_n = h(g_1\boldsymbol{E}_n\ ,\ g_2\boldsymbol{E}_n\ ,\ \ldots\ ,\ g_m\boldsymbol{E}_n) = h\boldsymbol{E}_m\boldsymbol{P}_m(g_1\boldsymbol{E}_n\ ,\ g_2\boldsymbol{E}_n\ ,\ \ldots\ ,\ g_m\boldsymbol{E}_n).$$

This is a member of **P** by the first of our preliminary result as above.

(7) Finally, suppose that f is obtained by minimalisation, $f(\boldsymbol{x}) = \min_y\{g(y,\boldsymbol{x}) = 0\}$ say, where $f : \mathbb{N}^n \to \mathbb{N}$ and $g : \mathbb{N}^{n+1} \to \mathbb{N}$ and $g\boldsymbol{E}_{n+1}$ is a member of **P**. But then so is the function

$$x \mapsto \min_y\{g\boldsymbol{E}_{n+1}P(y,x) = 0\} = \min_y\{g(y,\boldsymbol{E}_n(x)) = 0\} = f\boldsymbol{E}_n(x)$$ ■

E.2 Theorem

There is a partial recursive function $\varphi : \mathbb{N}^2 \to \mathbb{N}$ such that the functions

$$\varphi(0,\_)\ ,\ \varphi(1,\_)\ ,\ \varphi(2,\_)\ ,\ \ldots$$

are all the partial recursive functions $\mathbb{N} \to \mathbb{N}$. That is, given any partial recursive function $f : \mathbb{N} \to \mathbb{N}$, there is an i such that $f(x) = \varphi(i,x)$ for all x.

Note It is usual to write this function $\varphi_i(x)$ instead of $\varphi(i,x)$. One can then think of φ_0 , φ_1 , φ_2 , ... as being a listing of all the partial recursive functions $\mathbb{N} \to \mathbb{N}$.

Proof. Define φ_n thus:

Firstly $\varphi_1, \varphi_2, \ldots, \varphi_5$ are the six basic functions listed in the lemma above.

Then, for $n \geq 6$,

If $n = 3m + 6$ then φ_n is the function got from $\varphi_{L(m)}$ and $\varphi_{R(m)}$ by composition.

If $n = 3m + 7$ then φ_n is the function got from $g = \varphi_{L(m)}$ and $h = \varphi_{R(m)}$ by $\varphi_n(x) = P(g(x), h(x))$.

If $n = 3m + 8$ then φ_n is the function got from $g = \varphi_m$ by $\varphi_n(x) = \min_y\{g(P(y,x)) = 0\}$.

It is clear, from the lemma, that this will indeed list all the partially recursive functions $\mathbb{N} \to \mathbb{N}$. It remains to show that φ is itself partial recursive, as a function $\mathbb{N}^2 \to \mathbb{N}$. To do this we write a program.

```
1  main F(n,x; m,r,v) {
2      if (n=0) {return x+1};
3      if (n=1) {return L(x)};
4      if (n=2) {return R(x)};
5      if (n=3) {return L(x)+R(x)};
```

```
6       if (n=4) {return L(x)∸R(x)};
7       if (n=5) {return L(x)*R(x)};
8       if (n>5) {
9           m := ⌊n/3⌋ - 2;
10          r := Rem(m,3);
11          if (r=0){return F(L(m),F(R(m),x))};
12          if (r=1){return P(F(L(m),x),F(R(m),x))}
13          if (r=2) {
14              v := 0;
15              while (F(m,P(v,x))≠0) {v := v+1};
16              return v
17          }
18      };
19      return 0
20  }
```

■

E.3 Remarks

We now have a "gold-plated" effective listing of all partial recursive functions $\mathbb{N} \to \mathbb{N}$,

$$\varphi_0\,,\,\varphi_1\,,\,\varphi_2\,,\,\cdots$$

Using the recursive bijections $P_m : \mathbb{N}^m \to \mathbb{N}$ we can list the partial recursive functions $\mathbb{N}^m \to \mathbb{N}$

$$\varphi_0^{(m)}\,,\,\varphi_1^{(m)}\,,\,\varphi_2^{(m)}\,,\,\cdots$$

simply by defining $\varphi_i^{(m)} = \varphi_i P_m$. In fact we can list the partial recursive functions $\mathbb{N}^m \to \mathbb{N}^n$

$$\varphi_0^{(n,m)}\,,\,\varphi_1^{(n,m)}\,,\,\varphi_2^{(n,m)}\,,\,\cdots$$

just as simply, by defining $\varphi_i^{(n,m)} = \boldsymbol{E}_n \varphi_i P_m$.

In the particular ordering given by the chosen proof above, $\varphi_1, \varphi_2, \ldots, \varphi_5$ are the functions

$$\begin{aligned}
\varphi_0(x) &= x+1 \\
\varphi_1(x) &= L(x) \\
\varphi_2(x) &= R(x) \\
\varphi_3(x) &= L(x)+R(x) \\
\varphi_4(x) &= L(x) \dot{-} R(x) \\
\varphi_5(x) &= L(x)R(x)\ .
\end{aligned}$$

Also, for any i, j and x,

$$\begin{aligned}
\varphi_i\varphi_j(x) &= \varphi_{3P(i,j)+6}(x) \\
P(\varphi_i(x), \varphi_j(x)) &= \varphi_{3P(i,j)+7}(x)
\end{aligned}$$

and

$$\min_y\{\varphi_i(P(y,x))=0\} \;=\; \varphi_{3i+8}(x)\ .$$

► E.1 We notice that, as a consequence of the proof of Lemma E.1, there is a recursive function $\chi:\mathbb{N}^2\to\mathbb{N}$ such that

$$\varphi_j\varphi_i \;=\; \varphi_{\chi(j,i)} \qquad \text{for all } i \text{ and } j$$

(here $\varphi_j\varphi_i$ denotes composition).

More generally, because of the definitions just made, if we have

$$\mathbb{N}^l \xrightarrow{\varphi_i} \mathbb{N}^m \xrightarrow{\varphi_j} \mathbb{N}^n$$

then the indices compose by the same function: $\varphi_j^{(n,m)}\varphi_i^{(m,l)}=\varphi_{\chi(j,i)}^{(n,l)}$.

Another point worth noting is that, because of the definition of P_n,

$$P_{m+n}(x_1,x_2,\ldots,x_m,y_1,y_2,\ldots,y_n) \;=\; P_{m+1}(x_1,x_2,\ldots,x_m,P_n(y_1,y_2,\ldots,y_n))$$

and so

$$\varphi_i^{(m+n)}(x_1,x_2,\ldots,x_m,\boldsymbol{y}) \;=\; \varphi_i^{m+1}(x_1,x_2,\ldots,x_m,P_n(\boldsymbol{y}))\ .$$

In what follows it will be convenient to assume that we have chosen the particular listing of partial recursive functions given in the theorem. However it is not necessary to do this: most of the results still hold whatever "effective" listing is used (but then some of the proofs must become more general).

One should observe that the listing is not one-to-one, in fact it is easy to see that any partial recursive function appears in the list infinitely often. Less obvious is the fact that this must be so in *any* effective listing: it is just not possible to effectively list the partial recursive functions in such a way that every function appears just once. This explains some of the ► F.8 choices of words in the theorems which follow. Rice's Theorem (F.8 below) shows just how bad the situation must be in this regard.

E.4 Lemma

For every partial recursive function $f:\mathbb{N}^n\to\mathbb{N}$ there is a recursive function $\hat{f}:\mathbb{N}^n\to\mathbb{N}$ (not merely partial recursive!) such that

$$f\circ(\varphi_{i_1},\varphi_{i_2},\ldots,\varphi_{i_n}) \;=\; \varphi_{\hat{f}(i_1,i_2,\ldots,i_n)} \qquad \text{for all } i_1,i_2,\ldots,i_n.$$

(The function on the left is $x \;\mapsto\; f(\varphi_{i_1}(x),\varphi_{i_2}(x),\ldots,\varphi_{i_n}(x))$. This is the notation introduced in Subsection A.4.) ► A.4

► A.2 **Proof.** is by induction over the construction of f as given in Definition A.2.

Suppose first that f is the successor function, $f(x) = x + 1$. Then

$$f(\varphi_i(x)) \;=\; \varphi_i(x) + 1 \;=\; \varphi_0\varphi_i(x) \;=\; \varphi_{\chi(0,i)}(x)$$

so the result is true with $\hat{f}(i) = \chi(0, i)$.

Suppose next that f is the projection function $f \circ (x_1, x_2, \ldots, x_n) = x_k$. Then

$$f \circ (\varphi_{i_1}, \varphi_{i_2}, \ldots, \varphi_{i_n}) \;=\; \varphi_{i_k} \;=\; \varphi_{f(i_1,i_2,\ldots,i_n)}$$

so in this case $\hat{f} = f$.

Suppose next that f is addition, $f(x, y) = x + y : \mathbb{N}^2 \to \mathbb{N}$. Then

$$fE_2(w) \;=\; f(L(w), R(w)) \;=\; L(w) + R(w) \quad \text{so} \quad f \;=\; \varphi_3^{(2)}.$$

also

$$f \circ (\varphi_i, \varphi_j) \;=\; fE_2P \circ (\varphi_i, \varphi_j) \;=\; \varphi_3\varphi_{3P(i,j)+7} \;=\; \varphi_{\chi(3,3P(i,j)+7)}$$

so the result is true with $\hat{f} = \chi(3, 3P(i, j) + 7)$.

The same argument holds if f is natural subtraction or multiplication.

Suppose now that $f = P : \mathbb{N}^2 \to \mathbb{N}$; then $f \circ (\varphi_i, \varphi_j) = P \circ (\varphi_i, \varphi_j) = \varphi_{3P(i,j)+7}$ and so the result is true with $\hat{f} = 3P(i, j) + 7$. Using this, we prove the lemma in the case $f = P_n : \mathbb{N}^n \to \mathbb{N}$ by induction over n. If $n = 1$ then $f = P_1 = \mathrm{id}$ so $f(\varphi_i) = \varphi_i$, and so $\hat{f}$ is the identity also, $\hat{f} = \varphi_d$, where d is the index of the identity function. Now, assuming the result is true for n,

$$\begin{aligned} P_{n+1} \circ (\varphi_{i_1}, \varphi_{i_2}, \ldots, \varphi_{i_{n+1}}) \;&=\; P \circ (\varphi_{i_1}, P_n(\varphi_{i_2}, \varphi_{i_3}, \ldots, \varphi_{i_{n+1}})) \\ &=\; P \circ (\varphi_{i_1}, \varphi_{\hat{P}_n(i_2,i_3,\ldots,i_{n+1})}) \qquad \text{by the inductive hypothesis} \\ &=\; \varphi_{3P(i_1,\hat{P}_n(i_2,i_3,\ldots,i_{n+1}))+7} \end{aligned}$$

and so the result is true with $\hat{f} = 3P(i_1, \hat{P}_n(i_2, i_3, \ldots, i_{n+1})) + 7$.

Suppose now that f is given by substitution, $f = h \circ (g_1, g_2, \ldots, g_m) : \mathbb{N}^n \to \mathbb{N}$. Then

$$\begin{aligned} f \circ (\varphi_{i_1},&\varphi_{i_2}, \ldots, \varphi_{i_n}) \;= \\ &=\; h \circ (g_1 \circ (\varphi_{i_1}, \varphi_{i_2}, \ldots, \varphi_{i_n}), g_2 \circ (\varphi_{i_1}, \varphi_{i_2}, \ldots, \varphi_{i_n}), \ldots, g_m \circ (\varphi_{i_1}, \varphi_{i_2}, \ldots, \varphi_{i_n})) \\ &=\; h \circ (\varphi_{\hat{g}_1(i_1,i_2,\ldots,i_n)}, \varphi_{\hat{g}_2(i_1,i_2,\ldots,i_n)}, \ldots, \varphi_{\hat{g}_m(i_1,i_2,\ldots,i_n)}) \\ &=\; \varphi_{\hat{h}(\hat{g}_1(i_1,i_2,\ldots,i_n),\hat{g}_2(i_1,i_2,\ldots,i_n),\ldots,\hat{g}_m(i_1,i_2,\ldots,i_n))} \end{aligned}$$

so $\hat{f} = \hat{h} \circ (\hat{g}_1, \hat{g}_2, \ldots, \hat{g}_m)$.

Finally, suppose that f is given by minimalisation

$$f(x_1, x_2, \ldots, x_n) \;=\; \min_y\{g(y, x_1, x_2, \ldots, x_n) = 0\} \ .$$

Then, putting $w = P(y,x)$,

$$\begin{aligned} g(y,\varphi_{i_1}(x),\varphi_{i_2}(x),\ldots,\varphi_{i_n}(x)) &= g(L(w),\varphi_{i_1}R(w),\varphi_{i_2}R(w),\ldots,\varphi_{i_n}R(w)) \\ &= g(\varphi_1(w),\varphi_{i_1}\varphi_2(w),\varphi_{i_2}\varphi_2(w),\ldots,\varphi_{i_n}\varphi_2(w)) \\ &\qquad\qquad \text{since } L=\varphi_1 \text{ and } R=\varphi_2 \\ &= g(\varphi_1(w),\varphi_{\chi(i_1,2)}(w),\varphi_{\chi(i_2,2)}(w),\ldots,\varphi_{\chi(i_n,2)}(w)) \\ &= \varphi_\theta(w) = \varphi_\theta P(x,y) \end{aligned}$$

where $\theta = \hat{g}(1,\chi(i_1,2),\chi(i_2,2),\ldots,\chi(i_n,2)))$. So

$$f(\varphi_{i_1}(x),\varphi_{i_2}(x),\ldots,\varphi_{i_n}(x)) = \min_y\{\varphi_\theta P(x,y)=0\} = \varphi_{3\theta+8}$$

and the result is true again with $\hat{f} = 3\hat{g}(1,\chi(i_1,2),\chi(i_2,2),\ldots,\chi(i_n,2))) + 8$. ■

E.5 Lemma

For any partial recursive function $f : \mathbb{N}^2 \to \mathbb{N}$ there is a recursive function $\tilde{f} : \mathbb{N} \to \mathbb{N}$ such that $f(x,y) = \varphi_{\tilde{f}(x)}(y)$ for all x and y.

Proof. Consider first the case of the function $\pi(x,y) = x$ for all x and y. Define $\tilde{\pi}$ inductively:

$$\begin{aligned} \tilde{\pi}(0) &= z && \text{where } z \text{ is the index of the zero function: } \varphi_z(y)=0 \text{ for all } y, \\ \tilde{\pi}(x+1) &= \chi(0,\tilde{\pi}(x)) && \text{where } \chi \text{ is the composition function defijned above.} \end{aligned}$$

Let f be any partial recursive function $\mathbb{N}^2 \to \mathbb{N}$. Then, by the previous lemma, there is a recursive function $\hat{f} : \mathbb{N}^2 \to \mathbb{N}$ such that $\varphi_{\tilde{f}(i,j)}(y) = f(\varphi_i(y),\varphi_j(y))$ for all i,j and y. Set $\tilde{f} = \hat{f}(\tilde{\pi}(x),d)$, where d is the index of the identity function, $\varphi_d(y) = y$. Then

$$\varphi_{\tilde{f}(x)}(y) = \varphi_{\hat{f}(\tilde{\pi}(x),d)}(y) = f(\varphi_{\tilde{\pi}(x)}(y),\varphi_d(y)),= f(x,y)$$

as required. ■

E.6 The "s-m-n Theorem"

For each $m \geq 1$ and $n \geq 1$ there is a recursive function $s_{m,n} : \mathbb{N}^{m+1} \to \mathbb{N}$ such that

$$\varphi_i^{(m+n)}(\boldsymbol{x},\boldsymbol{y}) = \varphi_{s_{m,n}(i,\boldsymbol{x})}(\boldsymbol{y}) \qquad \text{for all } i \in \mathbb{N},\ \boldsymbol{x} \in \mathbb{N}^m \text{ and } \boldsymbol{y} \in \mathbb{N}^n.$$

Proof. We first prove the result in the case $m = n = 1$; the general case then follows easily. So we prove: There is a recursive function $s : \mathbb{N}^2 \to \mathbb{N}$ such that

$$\varphi_i^{(2)}(x,y) = \varphi_{s(i,x)}(y) \qquad \text{for all } i,x,y \in \mathbb{N}.$$

We show how to define s recursively by giving a description of how to write a program for s (with comments).

E.1 If $i \leq 5$ then φ_i is one of the six basic functions listed in Lemma E.1. Therefore $\varphi_i^{(2)} = \varphi_i P$ is one of the functions

$$x \mapsto P(x,y)+1 \quad , \quad x \quad , \quad y \quad , \quad x+y \quad , \quad x \dot{-} y \quad \text{or} \quad xy.$$

Whichever it is, call it f. Set $s(i,x) = \tilde{f}(x)$ (where $\tilde{f}$ is the function defined in the preceding lemma) and then $\varphi_{s(i,x)}(y) = \varphi_{\tilde{f}(x)}(y) = f(x,y)$.

Otherwise $i \geq 6$. Write $m = \lfloor (i-6)/3 \rfloor$, so that $i = 3m+6, 3m+7$ 0r $3m+8$.

If $i = 3m+6$ then $\varphi_i = \varphi_{L(m)}\varphi_{R(m)}$ and so

$$\begin{aligned}
\varphi_i^{(2)}(x,y) &= \varphi_i P(x,y) \\
&= \varphi_{L(m)}\varphi_{R(m)}P(x,y) \\
&= \varphi_{L(m)}\varphi^{(2)}_{R(m)}(x,y) \\
&= \varphi_{L(m)}\varphi_{s(R(m),x)}(y) \\
&= \varphi_{\chi(L(m),s(R(m),x))}(y)
\end{aligned}$$

and so we set $s(i,x) = \chi(L(m), s(R(m),x))$.

If $i = 3m+7$ then $\varphi_i = P(\varphi_{L(m)}, \varphi_{R(m)}$ and so

$$\begin{aligned}
\varphi_i^{(2)}(x,y) &= \varphi_i P(x,y) \\
&= P(\varphi_{L(m)}P(x,y), \varphi_{R(m)}P(x,y)) \\
&= P(\varphi^{(2)}_{L(m)}(x,y), \varphi^{(2)}_{R(m)}(x,y)) \\
&= P(\varphi_{s(L(m),x)}(y), \varphi_{s(R(m),x)}(y)) \\
&= \varphi_{3k+7}(y)
\end{aligned}$$

where $k = P(s(L(m),x), s(R(m),x))$, so we set $s(i,x) = 3P(s(L(m),x), s(R(m),x)) + 7$.

Before moving on, note that the function $\mathbb{N}^3 \to \mathbb{N}^3$ given by $(z,x,y) \mapsto (x,z,y)$ is recursive, so let its index be r: $\varphi_r^{(3,3)}(z,x,y) = (x,z,y)$. Then, defining $\rho : \mathbb{N} \to \mathbb{N}$ by $\rho(i) = \chi(i,r)$, we have $\varphi_{\rho(i)}(z,x,y) = \varphi_i(x,z,y)$ for all i,x,y and z.

If $i = 3m+8$ then $\varphi_i(w) = \min_z\{\varphi_m P(z,w) = 0\}$ for all w. Then

$$\begin{aligned}
\varphi_i^{(2)}(x,y) &= \varphi_i P(x,y) \\
&= \min_z\{\varphi_m P(z, P(x,y)) = 0\} \\
&= \min_z\{\varphi^{(3)}_m(z,x,y) = 0\} \\
&= \min_z\{\varphi_{\rho(m)}(x,z,y) = 0\} \\
&= \min_z\{\varphi^{(2)}_{\rho(m)}(x, P(z,y)) = 0\} \\
&= \min_z\{\varphi_{s(\rho(m),x)}P(z,y) = 0\} \\
&= \varphi_{3s(\rho(m),x)+8}(y)
\end{aligned}$$

so set $s(i,x) = 3s(\rho(m),x) + 8$.

This completes the proof when $m = n = 1$.

Now we prove the result for $m = 1$ and any n. We want to prove that, for any $n \geq 1$, there is a recursive function $s_{1,n} : \mathbb{N}^2 \to \mathbb{N}$ such that $\varphi_i^{(n+1)}(x, \boldsymbol{y}) = \varphi_{s_{1,n}(i,x)}^{(n)}(\boldsymbol{y})$ for all $i, x \in \mathbb{N}$ and $\boldsymbol{y} \in \mathbb{N}^n$. This is easy: the function s already defined will do:—

$$\varphi_i^{(n+1)}(x, \boldsymbol{y}) \;=\; \varphi_i^{(2)}(x, P_n(\boldsymbol{y})) \;=\; \varphi_{s(i,x)} P_n(\boldsymbol{y}) \;=\; \varphi_{s(i,x)}^{(n)}(\boldsymbol{y}) \; .$$

Finally we prove the result for any $m \geq 1$ and $n \geq 1$, by induction over m. The case $m = 1$ has been proved above. Then, for the case $m + 1$,

$$\begin{aligned} \varphi_i^{(m+n+1)}(x_1, x_2, \ldots, x_{m+1}, \boldsymbol{y}) \;&=\; \varphi_{s(i,x_1)}^{(m+n)}(x_2, x_3, \ldots, x_{m+1}, \boldsymbol{y}) \qquad \text{just proved above,} \\ &=\; \varphi_{s_{m,n}(s(i,x_1),(x_2,x_3,\ldots,x_{m+1}))}(\boldsymbol{y}) \end{aligned}$$

and so the result is true with $s_{m+1,n}(i, x_1, x_2, \ldots, x_{m+1}) = s_{m,n}(s(i, x_1), (x_2, x_3, \ldots, x_{m+1}))$. ■

E.7 Remark

▶ E.2

It was remarked above that the listing of partial recursive functions given by Theorem E.2 is by no means the only possible one. Virtually all useful results about such effective listings can be proved using the two main theorems of this section (Theorem E2, as an existence theorem, and the s-m-n Theorem). Thus we can take these two theorems as a definition of what an "effective" listing is.

F Some important negative results

In this section I will head each theorem, in italics, with an informal description of what it tells us.

F.1 Theorem

There is no universal recursive function.

There is no recursive function $\psi : \mathbb{N}^2 \to \mathbb{N}$ such that the functions

$$\psi(0, \_) \quad , \quad \psi(1, \_) \quad , \quad \psi(2, \_) \quad , \quad \dots$$

are all the recursive functions $\mathbb{N} \to \mathbb{N}$ (i.e. such that, for every recursive function $f : \mathbb{N} \to \mathbb{N}$, there is some $i \in \mathbb{N}$ such that $f(x) = \psi(i, x)$ for all x.

Proof. Suppose that ψ exists. Define $g : \mathbb{N} \to \mathbb{N}$ by

$$g(x) \;=\; \psi(x, x) + 1 \qquad\qquad \text{for all } x \in \mathbb{N}.$$

Then g is recursive, so there is some $i \in \mathbb{N}$ such that $g = \psi(i, \_)$, that is,

$$g(x) \;=\; \psi(i, x) \qquad\qquad \text{for all } x \in \mathbb{N}.$$

In the case $x = i$, the last two displayed equations give $\psi(i, i) = g(i) = \psi(i, i) + 1$, a contradiction. ■

One of the points of Theorem E.2 is that it shows how, given an algorithm for a partially recursive function f, to find a subscript i such that $f = \varphi_i$; and conversely, given i to find an algorithm for f. Thus we can think of the subscript i as a convenient way of encapsulating the algorithm for φ_i. So one way of asking what we can tell about a partial recursive function from a given algorithm to compute it is to ask what we can tell about the *function* φ_i from a knowledge of the *index* i. As we will see, the answer is "pretty well nothing". ▶ E.2

In this context note that the fact that, for any given partial recursive function f, there are an infinite number of algorithms to compute it, corresponds to the fact that there are an infinite number of indices i such that $\varphi_i = f$. Also note that, to ask whether such-and-such a question about a partial recursive function can be answered from a knowledge of a given algorithm to compute it is to ask whether the answer to the question about φ_i is a recursive predicate of the subscript i. The next Theorem answers an obvious such question in the negative.

F.2 Theorem

Given an algorithm for a partial recursive function, one cannot in general determine whether it is recursive or not.

The set $\{\, i : \varphi_i \text{ is recursive} \,\}$ is not recursive.

Proof. Suppose that this set is recursive. Then its characterisic function

$$C(i) = \begin{cases} 1 & \text{if } \varphi_i \text{ is recursive} \\ 0 & \text{otherwise} \end{cases}$$

is recursive. Define $\psi : \mathbb{N}^2 \to \mathbb{N}$ by

$$\psi(i,x) = \begin{cases} \varphi_i(x) & \text{if } C(i) = 1 \\ 0 & \text{otherwise} \end{cases}$$

▶ F.1 and we have a function which contradicts Theorem F.1. ■

F.3 Theorem

The Halting Problem is not recursively solvable.

The set dom φ is not recursive.

Proof. Suppose that dom φ is recursive. Then so is its characteristic function

$$C(i,x) = \begin{cases} 1 & \text{if } \varphi(i,x) \text{ is defined} \\ 0 & \text{otherwise.} \end{cases}$$

Now define $\psi : \mathbb{N}^2 \to \mathbb{N}$ by

$$\psi(i,x) = \begin{cases} \varphi(i,x) & \text{if } C(i,x) = 1 \\ 0 & \text{otherwise} \end{cases}$$

▶ F.1 and we have another function which contradicts Theorem F.1. ■

F.4 Theorem

Worse. There is a particular partial recursive function f whose halting problem is not solvable.

Proof. There are in fact many. I will give two examples.

Define the function f by $f = \varphi(L, R)$, that is, $f(x) = \varphi(L(x), R(x))$ for all x. Suppose that dom f is recursive. Then so is its characteristic function,

$$C(x) \;=\; \begin{cases} 1 & \text{if } \varphi(L(x), R(x)) \text{ is defined} \\ 0 & \text{otherwise.} \end{cases}$$

Define $\psi : \mathbb{N}^2 \to \mathbb{N}$ by

$$\psi(i,x) \;=\; \begin{cases} \varphi(i,x) & \text{if } C(P(i,x)) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

This is the ψ from the last theorem, so dom f is not recursive.

For a second example, define $f(x) = \varphi(x, x)$. Suppose that dom f is recursive, with characteristc function C . Define $g : \mathbb{N} \to \mathbb{N}$ by

$$g(x) \;=\; \begin{cases} \varphi(x,x)+1 & \text{if } C(x) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

Then g is recursive. So there is $i \in \mathbb{N}$ such that $g = \varphi_i$, that is, $g(x) = \varphi(i, x)$ for all x. Put $x = i$. Noting that $\varphi(i, i) = g(i)$, which is defined, so that $i \in$ dom f and $C(i) = 1$, we have the contradiction

$$\varphi(i,i) \;=\; g(i) \;=\; \varphi(i,i)+1 \;. \qquad \blacksquare$$

F.5 Theorem

You cannot tell if any particular number occurs as a value either.

Given any $y \in \mathbb{N}$, the set $S = \{\, (i, x) : \varphi(i, x) = y \,\}$ is not recursive.

Proof. Suppose that S is recursive. Then we will solve the Halting Problem. Define $\psi : \mathbb{N}^2 \to \mathbb{N}$ by $\psi(i, x) = \varphi(i, x) \dot{-} \varphi(i, x) + y$. Then

$$\psi(i,x) \;=\; \begin{cases} y & \text{if } (i,x) \in \text{dom } \varphi \\ \text{undefined} & \text{otherwise} \end{cases}$$

and (by Lemma E.5) there is recursive $\alpha : \mathbb{N} \to \mathbb{N}$ such that $\psi(i, x) = \varphi(\alpha(i), x)$ for all i, x. ▶ E.5 Then

$$(i,x) \in \text{dom } \varphi \quad\Leftrightarrow\quad \psi(i,x) = y \quad\Leftrightarrow\quad \varphi(\alpha(i),x) = y \quad\Leftrightarrow\quad (\alpha(i),x) \in S \;.$$

The last set here is recursive and the first set is not. $\blacksquare$

To see why the last set here is recursive, see Remark (ii) below. To see how Lemma E.5 is applied, use it in the case $n = 1$, with $f(u) = u \dot{-} u + y$ and $\alpha = \tilde{f}$. ▶ E.5

F.6 Remark

Suppose that R is a recursive subset of $\mathbb{N}^n$.

(i) Then so is $\mathbb{N} \setminus R$.

(ii) Let the function $h : \mathbb{N}^m \to \mathbb{N}^n$ be recursive also. Then so is the set $\{\, x : h(x) \in R \,\} = h^{-1}[R]$.

Proof. Let C be the characteristic function of R. Then the characteristic function of $\mathbb{N}^n \setminus R$ is $1 \dot{-} C$ and the characteristic function of $h^{-1}[R]$ is $C \circ h$. $\blacksquare$

F.7 Theorem

There is no effective procedure to decide what algorithm gives what function.

Let g be any partial recursive function $\mathbb{N} \to \mathbb{N}$. Then the set $\{\, i : \varphi_i = g \,\}$ is not recursive.

Proof. Suppose first that $g \neq \varnothing$, i.e. that g has nonempty domain. Let θ be a partial recursive function $\mathbb{N} \to \mathbb{N}$ with nonrecursive domain (e.g. one of the examples of Theorem F.4).

▶ F.4

Suppose that the set $R = \{\, i : \varphi_i = g \,\}$ is recursive.

Define a new function $f : \mathbb{N}^2 \to \mathbb{N}$ by $f(u, x) = \theta(u) \dot{-} \theta(u) + g(x)$. This is partial recursive and is the function

$$f(u,x) \;=\; \begin{cases} g(x) & \text{if } u \in \operatorname{dom} \theta \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Now, by Lemma E5 there is a *recursive* $\alpha : \mathbb{N} \to \mathbb{N}$ such that $f(u, x) = \varphi_{\alpha(u)}(x)$ for all u, x. Thus

$$\varphi_{\alpha(u)}(x) \;=\; \begin{cases} g(x) & \text{if } u \in \operatorname{dom} \theta \\ \text{undefined} & \text{otherwise,} \end{cases}$$

which is to say

$$\varphi_{\alpha(u)} \;=\; \begin{cases} g & \text{if } u \in \operatorname{dom} \theta \\ \varnothing & \text{otherwise.} \end{cases}$$

Therefore (and here is where we use $g \neq \varnothing$),

$$u \in \operatorname{dom} \varphi \quad\Leftrightarrow\quad \varphi_{\alpha(u)} = g \quad\Leftrightarrow\quad \alpha(u) \in R \quad\Leftrightarrow\quad u \in \alpha^{-1}[R].$$

This is a contradiction, since $\alpha^{-1}[R]$ is recursive but dom φ is not.

It remains to show that the set $\{\, i : \varphi_i = \varnothing \,\}$ is not recursive.

Choose any nonempty partial recursive function $g : \mathbb{N} \to \mathbb{N}$ you like (say, the zero function). Now define f and α as above, so again we have

$$\varphi_{\alpha(u)} \;=\; \begin{cases} g & \text{if } u \in \operatorname{dom} \theta \\ \varnothing & \text{otherwise.} \end{cases}$$

Then

$$u \in \operatorname{dom} \varphi \quad\Leftrightarrow\quad \varphi_{\alpha(u)} \neq \varnothing \quad\Leftrightarrow\quad \alpha(u) \notin R \quad\Leftrightarrow\quad u \in \alpha^{-1}[\mathbb{N} \setminus R]$$

and we have a contradiction as before. ■

A special case of this theorem is that the set $\{\, i : \varphi_i = \varnothing \,\}$ is not recursive. This says that there is no way of telling, from an algorithm, whether or not its function has any values at all.

F.8 Rice's Theorem

And if you thought things were bad so far, try this theorem and its corollary.

Let R be a recursive subset of with the property that

$$\text{if} \quad i \in R \quad \text{and} \quad \varphi_i = \varphi_j \quad \text{then} \quad j \in R \quad \text{also.}$$

Then $R = \varnothing$ or $R = \mathbb{N}$.

Proof. We will suppose that R is neither $\varnothing$ nor $\mathbb{N}$ and deduce a contradiction.

First note that the complement $\mathbb{N} \setminus R$ has the same properties: it is recursive and, if $i \in \mathbb{N} \setminus R$ and $\varphi_i = \varphi_j$ then $j \in \mathbb{N} \setminus R$ also.

Let θ be a partial recursive function $\mathbb{N} \to \mathbb{N}$ with nonrecursive domain (e.g. one of the examples of Theorem F.4). ▶ F.4

Now the empty function is partial recursive, so there is an index z such that $\varphi_z = \varnothing$. It is also either a member of R or its complement. We may suppose without loss of generality that $z \in \mathbb{N} \setminus R$. Since R is nonempty, it contains at least one number, k say, and then $\varphi_k \neq \varphi_z$. Summary so far:—

$$k \in R \quad , \quad z \in \mathbb{N} \setminus R \quad , \quad \varphi_k \neq \varphi_z = \varnothing \ .$$

Now define $f : \mathbb{N}^2 \to \mathbb{N}$ by $f(u, x) = \theta(u) \dot{-} \theta(u) + \varphi_k(x)$. Then f is partial recursive and is the function

$$f(u,x) \ = \ \begin{cases} \varphi_k(x) & \text{if } u \in \operatorname{dom} \theta \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Now there is a *recursive* $\alpha : \mathbb{N} \to \mathbb{N}$ such that $f(u, x) = \varphi_{\alpha(u)}(x)$ for all u, x. Thus

$$\varphi_{\alpha(u)}(x) \ = \ \begin{cases} \varphi_k(x) & \text{if } u \in \operatorname{dom} \theta \\ \text{undefined} & \text{otherwise,} \end{cases}$$

which is to say

$$\varphi_{\alpha(u)} \ = \ \begin{cases} \varphi_k & \text{if } u \in \operatorname{dom} \theta \\ \varnothing = \varphi_z & \text{otherwise.} \end{cases}$$

Now

$$\text{if} \quad u \in \operatorname{dom} \theta \quad \text{then} \quad \varphi_{\alpha(u)} = \varphi_k \quad \text{and so} \quad \alpha(u) \in R$$

and

$$\text{if} \quad u \notin \operatorname{dom} \theta \quad \text{then} \quad \varphi_{\alpha(u)} = \varphi_z \quad \text{and so} \quad \alpha(u) \in \mathbb{N} \setminus R \ .$$

It follows that $\operatorname{dom} \theta = \{\, u : \alpha(u) \in R \,\}$, which is recursive. ■

F.9 Corollary

There is no way of telling, from its algorithm, whether a function has any nontrivial property at all.

Let P be any nontrivial property of functions $\mathbb{N} \to \mathbb{N}$ ("nontrivial" here means that there is at least one function which has the property and at least one which does not). Then the set $\{ i : \varphi_i \text{ has property } P \}$ is not recursive.

G Recursively enumerable sets

G.1 Definition

A subset of $\mathbb{N}$ is *recursively enumerable* (often abbreviated to *r.e.*) if it is the range of some partial recursive function $\mathbb{N} \to \mathbb{N}$.

G.2 Theorem

For a subset R of $\mathbb{N}$, the following are equivalent:

(i) R is recursively enumerable (i.e. the range of a partial recursive function $\mathbb{N} \to \mathbb{N}$).

(ii) R is empty or the range of a recursive function $\mathbb{N} \to \mathbb{N}$.

(iii) R is empty or the range of a primitive recursive function $\mathbb{N} \to \mathbb{N}$.

Proof. Clearly (iii) $\Rightarrow$ (ii) $\Rightarrow$ (i), so it remains to prove (i) $\Rightarrow$ (iii).

Suppose then that R is the range of a partial recursive function f and is nonempty; we show that it is the range of a primitive recursive function. Using Corollary C.5, there are primitive recursive functions p and $q : \mathbb{N}^2 \to \mathbb{N}$ such that ▶ C.5

$$f(\boldsymbol{x}) \;=\; p(\,\min_t\{q(t,\boldsymbol{x}) = 0\}\,,\,\boldsymbol{x}\,)\ .$$

Then the function

$$\bar{q}(t,\boldsymbol{x}) \;=\; \begin{cases} 1 & \text{if } t \text{ is a first zero of } q \\ 0 & \text{otherwise} \end{cases}$$

is primitive recursive also. Here by "a first zero" I mean that $q(t,\boldsymbol{x}) = 0$ but $q(t',\boldsymbol{x}) \neq 0$ for all $t' < t$. (In case it is not obvious that $\bar{q}$ is primitive recursive, it is proved at the end of the main part of this proof, so as not to halt the flow.)

From the definition of $\bar{q}$ we see that $\bar{q}(t,\boldsymbol{x}) = 1$ if and only if $t = \min_t\{q(t,\boldsymbol{x}) = 0\}$ and so

$$R \;=\; \{\, p(t,\boldsymbol{x}) : \bar{q}(t,x) = 1 \,\}\ .$$

Now R is nonempty, so there is some $n_0 \in R$. Using this, define a function $r : \mathbb{N}^2 \to \mathbb{N}$ by

$$r(t,\boldsymbol{x}) \;=\; \bar{q}(t,\boldsymbol{x})p(t,\boldsymbol{x}) + (1 \dot{-} \bar{q}(t,\boldsymbol{x}))n_0\ .$$

With this definition it is obvious that r is primitive recursive. But, since $\bar{q}$ only takes values 0 and 1, the definition may be restated

$$r(t,\boldsymbol{x}) \;=\; \begin{cases} p(t,\boldsymbol{x}) & \text{if } \bar{q}(t,\boldsymbol{x}) = 1 \\ n_0 & \text{otherwise.} \end{cases}$$

Then $R = \operatorname{ran} r$ with r primitive recursive. It remains to replace r by a primitive recursive function $\mathbb{N} \to \mathbb{N}$. But

$$g(u) \;=\; r(L(u), R(u))$$

will do the trick, because the functions L and R are primitive recursive.

Appendix To see that $\bar{q}$ is primitive recursive, we manufacture it from q using only operations that we know will lead from primitive recursive functions to primitive recursive functions. Given q, define q_1 by

$$q_1(t, \boldsymbol{x}) \;=\; \prod_{u=0}^{t} q(u, \boldsymbol{x}).$$

Now q_1 has the property that

$$q_1(t, \boldsymbol{x}) \;=\; \begin{cases} 0 & \text{if there is } u \le t \text{ such that } q(u, \boldsymbol{x}) = 0 \\ \text{nonzero} & \text{otherwise.} \end{cases}$$

Next define q_2 by

$$q_+2(t, \boldsymbol{x}) \;=\; 1 \dot{-} q_1(t, \boldsymbol{x}) \;.$$

Then q_2 is the function

$$q_2(t, \boldsymbol{x}) \;=\; \begin{cases} 1 & \text{if there is } u \le t \text{ such that } q(u, \boldsymbol{x}) = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Next define

$$q_3(t, \boldsymbol{x}) \;=\; \sum_{u=0}^{t} q_2(u, \boldsymbol{x})$$

and then

$$q_3(t, \boldsymbol{x}) \;=\; \begin{cases} 1 & \text{if } u \text{ is a first zero of } q \\ 0 \text{ or } \ge 2 & \text{otherwise.} \end{cases}$$

► A.3 Now set $\bar{q}(t, \boldsymbol{x}) = \mathrm{eq}(q_3(t, \boldsymbol{x}), 1)$. (Here eq is the Equality Test function defined in A.3(iv).)

$$q_1(t, \boldsymbol{x}) \;=\; \begin{cases} 0 & \text{if there is } u \le t \text{ such that } q(u, \boldsymbol{x}) = 0 \\ \text{nonzero} & \text{otherwise.} \end{cases}$$

■

G.3 Remark

Since we have primitive recursive one-to-one correspondences between $\mathbb{N}$ and $\mathbb{N}^n$ for each n, functions $\mathbb{N} \to \mathbb{N}$ can be replaced by functions $\mathbb{N}^n \to \mathbb{N}$ in the above theorem. Using the same idea, recursively enumerable subsets of $\mathbb{N}^n$ can be defined.

G.4 Theorem

A subset R of $\mathbb{N}$ is recursively enumerable if and only if it is the domain of a partial recursive function $\mathbb{N} \to \mathbb{N}$.

Proof. First, suppose that R is recursively enumerable. Then it is the range of a recursive function $\mathbb{N} \to \mathbb{N}$. Define $g : \mathbb{N} \to \mathbb{N}$ by

$$g(x) \;=\; \min_i \{f(i) = x\} \;.$$

Then R is the domain of g.

Conversely, suppose that R is the domain of a partially recursive function $g : \mathbb{N} \to \mathbb{N}$. Define $f : \mathbb{N} \to \mathbb{N}$ by

$$f(x) \;=\; x.\mathbf{1}(g(x))$$

where $\mathbf{1}$ is the constant function $t \mapsto 1$. Then $\mathbf{1}(g(x))$ is 1 if $g(x)$ is defined and undefined otherwise. Therefore R is the range of f. ■

G.5 Theorem

If a subset is recursive then it is recursively enumerable.

Proof. Let R be a recursive set, C its characteristic function. Then R is the domain of the partial recursive function

$$f(x) \;=\; \min_i\{C(x) = 1\}\ .$$ ■

G.6 Theorem

A subset R of $\mathbb{N}$ is recursive if and only if both it and its complement $\mathbb{N} \smallsetminus R$ are recursively enumerable.

Proof. If R is recursive then so is its complement and then they are both recursively enumerable by the previous theorem.

Now suppose both R and its complement are recursively enumerable. If either one is empty the result is trivial, so we will suppose they are both nonempty. Then there are recursive functions f and g such that R is the range of f and its complement is the range of g. Now define a new function $h : \mathbb{N} \to \mathbb{N}$ by

$$h(x) \;=\; \begin{cases} f(\lfloor \frac{x}{2} \rfloor) & \text{if } x \text{ is even} \\ g(\lfloor \frac{x \dot{-} 1}{2} \rfloor) & \text{otherwise.} \end{cases}$$

The point of this is that the sequence of values of h consists of alternate values of f and g:

$$\begin{array}{ll} h(0) = f(0) & h(1) = g(0) \\ h(2) = f(1) & h(3) = g(1) \\ h(4) = f(2) & h(5) = g(2) \\ h(6) = f(3) & h(7) = g(3) \end{array}$$

and so on. In particular, the set of even terms of h is R and the set of odd terms is its complement. Therefore the function

$$\theta(x) = \min_i\{h(i) = x\}$$

is recursive (total!) and has the property that

$$\theta(x) \text{ is even if } x \in R \qquad \text{and} \qquad \theta(x) \text{ is odd if } x \notin R\ .$$

Consequently, the characteristic function of R can be computed as

$$C(x) = \begin{cases} 1 & \text{if } \theta(x) \text{ is even} \\ 0 & \text{otherwise} \end{cases}$$

and this is recursive. ■

And just in case you've been wondering whether there are in fact any recursively enumerable sets which are not recursive, the answer is yes. In Section F we found several partial recursive functions whose domains were not recursive. These domains however must be recursively enumerable. Perhaps the most obvious example is the domain of φ.

▶ F

G.7 Theorem

Let A and B be recursively enumerable sets. Then so are $A \cup B$ and $A \cap B$.

Proof. If either A or B is empty the result is trivial, so now we may assume that A and B are the ranges of recursive functions f and g respectively. Define h as in the previous Theorem. Then h is recursive and $A \cup B$ is the range of h, so it is recursively enumerable.

A and B are also the domains of partial recursive functions a and b respectively and then $A \cap B$ is the domain of $a + b$. ■

Note that, if A is a recursively enumerable set, then its complement need not be. Indeed, if A is not actually recursive, then its complement cannot be recursively enumerable (by Theorem G.6 above).

▶ G.6

In the next chapter we will need a slight generalisation of Theorem G.6:

▶ G.6

G.8 Theorem

Let A and B be disjoint recursively enumerable sets whose union is recursive. Then they are both recursive.

Proof. By symmetry, it is enough to show that A is recursive. Since it is recursively enumerable, it is enough to show that $\mathbb{N} \setminus A$ is also recursively enumerable. Now $\mathbb{N} \setminus (A \cup B)$ is recursive and so recursively enumerable. Therefore $B \cup (\mathbb{N} \setminus (A \cup B))$ is also recursively enumerable. But this is the set $\mathbb{N} \setminus A$. ■

9. GÖDEL'S THEOREM

A Gödel numbering

It will be assumed throughout this chapter that we are dealing with a (fixed) first-order language with arithmetic, as discussed in Section **4**.B, and which I will call **S**. Such a language has the relation $=$ (equality, binary) and the functions $\bar{0}$ (zero, nullary), $x \mapsto x^+$ (successor, unary), $+$ and $\times$ (addition and multiplication, binary), but, since it is not necessarily **ETA** but any theory which *contains* arithmetic, it may contain other relations and functions. ▶ 4.B

Recall that, to any natural number ξ, we defined a corresponding term $\bar{\xi}$ in **S** so that

$$\bar{1} \ominus \bar{0}^+ \quad , \quad \bar{2} \ominus \bar{0}^{++} \quad , \quad \bar{3} \ominus \bar{0}^{+++}$$

and so on (see Definition **4**.B.3). ▶ 4.B.3

A.1 Numbering of sequences again

In Definition B.16 we had a numbering of all finite sequences ▶ B.16

$$\sigma \,:\, \bigcup_{n=0}^{\infty} \mathbb{N}^n \;\to\; \mathbb{N}\,.$$

which was very useful to us then. It is easy to see that the function σ is injective (because P and all the P_n are). It is also very nearly bijective (because P and all the P_n for $n \geq 1$ are). It fails to be bijective because there is only one sequence of length 0, so the range of P_0 is $\{0\}$, not $\mathbb{N}$.

In this Chapter it will be useful to have such a function that is bijective. Luckily we are not here interested in the empty sequence so, by removing it from the domain, we have a neat definition which is very nearly the same as the one above.

We define

$$\gamma \,:\, \bigcup_{n=1}^{\infty} \mathbb{N}^n \;\to\; \mathbb{N}\,.$$

as follows: for any sequence $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ of length $n \geq 1$,

$$\gamma(\mathbf{x}) \;=\; P(n-1\,,\, P_n(x_1, x_2, \ldots, x_n))\ .$$

Looking at the definitions in the previous section we see that we have an alternative way of expressing this definition: $\gamma(x_1, x_2, \ldots, x_n) \;=\; P_{n+1}(n-1, x_1, x_2, \ldots, x_n)$.

As before, it follows immediately from this definition that the restriction γ_n of γ to any $\mathbb{N}^n$,

$$\gamma_n \,:\, \mathbb{N}^n \to \mathbb{N}$$

is primitive recursive.

And now, because P and all the P_n for $n \geq 1$ are bijective, so is γ.

Unsurprisingly, the basic operations we can perform with this function are the same (see Lemma B.17).

▶ B.17

A.2 Lemma

There are primitive recursive functions len : $\mathbb{N} \to \mathbb{N}$, ent : $\mathbb{N}^2 \to \mathbb{N}$, del : $\mathbb{N} \to \mathbb{N}$, add : $\mathbb{N}^2 \to \mathbb{N}$, rep : $\mathbb{N}^3 \to \mathbb{N}$ with these properties:

(i) $\text{len}(w)$ gets the length of the sequence: if $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\gamma(\mathbf{x}) = w$ then $\text{len}(w) = n$.

(ii) $\text{ent}(i, w)$ gets the ith entry of the sequence: if $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, $\gamma(\mathbf{x}) = w$ and $1 \leq i \leq n$ then $\text{ent}(i, w) = x_i$.

(iii) $\text{del}(w)$ represents removing the first entry of the sequence: if $n \geq 1$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, $\mathbf{x}' = (x_2, x_3, \ldots, x_n)$, $\gamma(\mathbf{x}) = w$ and $\gamma(\mathbf{x}') = w'$ then $\text{del}(w) = w'$.

(iv) $\text{add}(z, w)$ represents adding a new first entry z to the sequence: if $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, $\mathbf{x}' = (z, x_1, x_2, \ldots, x_n)$, $\gamma(\mathbf{x}) = w$ and $\gamma(\mathbf{x}') = w'$ then $\text{add}(z, w) = w'$.

(v) $\text{rep}(r, w, y)$ represents replacing the rth entry of the sequence by the value y: if $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, $\mathbf{x}' = (x_1, x_2, \ldots, x_{r-1}, y, x_{r+1}, \ldots, x_n)$, $\gamma(\mathbf{x}) = w$ and $\gamma(\mathbf{x}') = w'$ then $\text{rep}(r, w, y) = w'$.

Proof. (i) This is trivial: $\text{len}(w) = L(w) + 1$.

(ii)–(v) The proofs for all the other cases are the same as those for Lemma B.17 — substitute γ for σ, otherwise they go word for word. ■

▶ B.17

A.3 Definition: Gödel numbering

The basic idea here is to assign numbers to various parts of the language **S** — to symbols, expressions and proofs — and so allow arithmetic to talk about itself to some extent. These numberings are called *Gödel numberings*.

First we number the symbols in our language. It does not matter much how we do this, provided that the numbers given to different symbols are different; we will assume that they are numbered in a one-to-one fashion. For example, suppose there are m (formal) function symbols, $f_1, f_2, \ldots, f_m$ of arities $\alpha_1, \alpha_2, \ldots, \alpha_m$ respectively, n (formal) relation symbols, $r_1, r_2, \ldots, r_n$ of arities $\beta_1, \beta_2, \ldots, \beta_m$ respectively and the variable symbols are $v_1, v_2, \ldots$; then we could number the symbols as follows:

$\neg$	0
$\Rightarrow$	1
$\forall$	2
(	3

)	4
,	5
$f_1, f_2, \ldots, f_m$	6,7,...,m+5
$r_1, r_2, \ldots, r_n$	m+6,m+7,...,m+n+5
$v_1, v_2, \ldots$	m+n+6,m+n+7,...

We may call this the *Gödel numbering of symbols* and denote it gnSym. Thus, for example, $\text{gnSym}(\forall) = 2$ and $\text{gnSym}(f_2) = 7$ (assuming f_2 actually exists).

We can now use this numbering to represent each string in the language by a sequence of natural numbers. For example, let us see how we would represent the axiom $(\forall x)(x^+ \neq \bar{0})$. Firstly, we have to rewrite this axiom in its fully formal form, according to the definitions of Chapter **3**; this is $(\forall x(\neg = (s(x), \bar{0})))$ (here I am using s for the successor function). Now we need to know what the Gödel numbers of the symbols $\bar{0}$, s, x and $=$ are; this would depend on how many extra functions and relations we have in our language **S**; let us suppose that

▶ Ch.3

$$\text{gnSym}(\bar{0}) = 7 \quad , \quad \text{gnSym}(s) = 8 \quad , \quad \text{gnSym}(=) = 10 \quad \text{and} \quad \text{gnSym}(x) = 11 \ .$$

Then our string would correspond to the sequence $(3, 2, 11, 3, 0, 10, 3, 8, 3, 11, 4, 5, 7, 4, 4, 4)$.

(So far, this is quite closely analogous to the way a computer deals with text. Each symbol of the text is stored as a number — its ASCII code — and words and expressions are stored as sequences of these numbers. But next we go further.)

In a formal language, none of the expressions are empty strings, so we need not deal with the empty string at all.

We can now use our γ numbering of nonempty sequences just defined above to make each string correspond to a single number, in the case of our example $\gamma(3, 2, 11, 3, 0, 10, 3, 8, 3, 11, 4, 5, 7, 4, 4, 4)$. This gives us a *Gödel numbering of strings*. More precisely, it gives us a function gnStr from the set of all (nonempty) strings in the language **S** to $\mathbb{N}$ defined thus: for the string $a_1 a_2 \ldots a_k$ (where $a_1, a_2, \ldots, a_k$ are the individual characters (symbols) in the string)

$$\text{gnStr}(a_1 a_2 \ldots a_k) \ = \ \gamma(\text{gnSym}(a_1), \text{gnSym}(a_2), \ldots, \text{gnSym}(a_n)) \ .$$

Since gnSym and γ are both bijective, so is gnStr and so any string in the language may be identified with its Gödel number. This allows us to make statements such as ...

A.4 Lemma

The set of all expressions in **S** is recursive.

By this I mean that the set of all string Gödel numbers of expressions in the language is a recursive subset of $\mathbb{N}$.

Informal proof. Consider the definition of an expression given in Chapter 3. This provides a completely mechanical method for checking whether any given string is an expression or not. Indeed, using the programming language described in Chapter 8, it is tedious but not difficult to write a program to compute the characteristic function of this set, thus showing that it is recursive.

▶ 3

▶ 8

Note Here and in what follows I will give a number of "informal proofs" that certain sets are recursive or recursively enumerable. A detailed proof would consist of the presentation of the appropriate algorithm. The algorithms for these proofs are indeed presented in Appendix C in case you should feel suspicious about any of the informal ones.

► C

A.5 Lemma

The set of all sentences in **S** is recursive.

Informal proof. As for the last lemma but notice also that deciding whether an expression contains any free variables or not is also a mechanical procedure.

A.6 Lemma

The set of all axioms of **PL** is recursive.

By this I mean of course that the set of all string Gödel numbers of axioms of **PL** is a recursive subset of $\mathbb{N}$.

Remark The six "axioms" of **PL** are of course axiom schemas and represent between them an infinite number of actual axioms. Basically they tell us that any expression with one of the six prescribed kind of structures is an axiom.

► 3

Informal proof. Consider the definition of an expression given in Chapter 3. This also provides a completely mechanical method for breaking up an expression into its component parts, so its structure can be compared with those of the six axiom schemas. Again it is tedious but not difficult to write a program to compute the characteristic function of this set, thus showing that it is recursive.

A.7 Remark: recursive axioms

Most first order systems use the six axiom schemas of **PL** plus a finite number of proper axioms. In this case of course their axioms form a recursive set. In cases where there are an infinite number of proper axioms, it is nearly always the case that these axioms also defined in such a way as to form a recursive set.

Thus we may suppose that the set of axioms of a first order system is usually recursive; a non-recursive set of axioms is a rather strange thing to have — consider, what is the use of a set of axioms if you have no way of deciding what is an axiom and what isn't?

Occasionally non-recursive sets of axioms are useful for theoretical reasons. For example, the set of axioms created in the proof of Theorem **3**.H.3 is (usually) non-recursive.

► **3**.H.3

At this point we want to introduce the idea that a theory may have alternative sets of axioms. We should get these ideas straight now.

Recall that in a deductive system, we have a **theory** which is the set of **theorems** which may be deduced from a chosen set of axioms. Given a set $\mathcal{A}$ of expressions, we write $\text{thm}(\mathcal{A})$ for the theory generated by $\mathcal{A}$, that is, the set of all expressions X such that $\mathcal{A} \vdash YX$. Thus we say that $\mathcal{A}$ is **a** set of axioms for the theory T if $T = \text{thm}(\mathcal{A})$. It may be the case that $\mathcal{A}$ is different from the axioms we chose to generate T in the first place.

We will say that a theory is **recursively axiomatisable** (or simply **axiomatisable**) if it has a recursive set of axioms (whether they were the ones chosen to generate it or not).

A.8 Remark: decidability

The idea of a theory being decidable – defined up to now somewhat informally by saying that there is an algorithm for deciding whether or not an expression is a theorem – may now be made precise: a theory is decidable if it (that is, the set of theorems) is recursive.

We know that, because of Universal Generalisation, an expression is a theorem if and only if its fully quantified form is a theorem. It follows that a theory is decidable if and only if the set of all sentences (closed expressions) which are theorems is recursive.

A.9 Gödel numbering continued

Proofs are sequences of expressions satisfying certain conditions. Thus they are sequences of strings (sequences of sequences if you like). It only takes one more step to form a numbering of such sequences of strings.

Suppose then that $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k$ is a sequence of strings. This corresponds in a bijective fashion to the sequence $(\text{gnStr}(\mathbf{s}_1), \text{gnStr}(\mathbf{s}_2), \ldots, \text{gnStr}(\mathbf{s}_k))$ of natural numbers which in turn corresponds to the single number $\gamma(\text{gnStr}(\mathbf{s}_1), \text{gnStr}(\mathbf{s}_2), \ldots, \text{gnStr}(\mathbf{s}_k))$.

This numbering of sequences of strings we will call the **sequence Gödel numbering** and denote it gnSeq. We will be interested in the sequences of strings which constitute proofs.

A.10 Lemma

In a recursively axiomatisable theory, the set of all proofs is recursive.

By this we mean of course that the set of all sequence Gödel numbers of proofs in the language is a recursive subset of $\mathbb{N}$.

Informal proof. Consider the definition of a proof given in Chapter **3**. This provides a completely mechanical method for checking whether any given sequence of strings is a proof or not. ▶ Ch.3

Consider what is involved. First we must step through the individual strings checking that each is indeed an expression: we recover their string Gödel numbers using the function ent, and test them as guaranteed by Lemma A.4 above. We must also check that each string arises by one of the rules for forming a proof – that it is a case of an axiom (checkable by assumption), that it follows from two earlier steps by MP or from one earlier step by UG. All these are mechanical things to check, though the algorithms might be tiresome to write out. ▶ A.4

And along the same lines:

A.11 Lemma

The "Proof of" relation

$\langle \xi, \eta \rangle \mapsto$ ξ is the number of some expression and η is the number of a proof of that expression

is recursive.

Informal proof. Same as for the previous lemma.

A.12 Theorem

In a recursively axiomatisable theory, the set of all theorems is recursively enumerable.

Proof. Let a be the string Gödel number of some fixed theorem, say the first instance of the first axiom of **PL**. Now consider the function $f : \mathbb{N} \to \mathbb{N}$ defined by the following algorithm: for each $w \in \mathbb{N}$, compute whether w is the Gödel number of a valid proof or not. If it is the Gödel number of a valid proof, output the string number of its last entry, $\mathrm{ent}(\mathrm{len}(w), w)$ — which is the string Gödel number of the theorem it proves. If w is not the Gödel number of a valid proof, then output a.

Then the set of all theorems is just the range of f. ■

A.13 Theorem

A recursively axiomatisable complete theory is decidable.

Proof. Let $\mathcal{T}$ be the set of all sentences (closed expressions) in the language which are theorems and let $\mathcal{U}$ be the set of all "untheorems" — sentences whose negation is a theorem (A is an untheorem means $\vdash \neg A$). The set of untheorems is clearly recursively enumerable too — simply enumerate the theorems as described above and negate each one as you go.

If $\mathcal{T}$ and $\mathcal{U}$ are not disjoint then the theory is inconsistent; then every expression is a theorem and so the set of theorems is recursive (the set of its Gödel numbers is $\mathbb{N}$).

Suppose now that $\mathcal{T}$ and $\mathcal{U}$ are disjoint. They are both recursively enumerable and their union is the set of all sentences (that is the definition of "complete"), which is recursive.
▶ **8**.G.8 Therefore (by **8**.G.8) $\mathcal{T}$ is recursive. ■

B Expressibility and representability

B.1 Definition

(i) An n-ary relation r on $\mathbb{N}$ is **expressible** in **S** if there is an expression $R(x_1, x_2, \ldots, x_n)$ in **S** with exactly n free variables $x_1, x_2, \ldots, x_n$ such that, for any $\xi_1, \xi_2, \ldots, \xi_n$ in $\mathbb{N}$

$$\begin{array}{lll} & \text{if } r(\xi_1, \xi_2, \ldots, \xi_n) \text{ is true} & \text{then } \vdash R(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n) \text{ in } \mathbf{S} \\ \text{and} & \text{if } r(\xi_1, \xi_2, \ldots, \xi_n) \text{ is false} & \text{then } \vdash \neg R(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n) \text{ in } \mathbf{S}. \end{array}$$

We then say the "R expresses r".

(ii) A function $f : \mathbb{N}^n \to \mathbb{N}$ is **representable** in **S** if there is an expression $F(x_1, x_2, \ldots, x_n, y)$ in **S** with exactly $n+1$ free variables $x_1, x_2, \ldots, x_n$ such that, for any $\xi_1, \xi_2, \ldots, \xi_n, \eta$ in $\mathbb{N}$,

$$\begin{array}{l} \text{if } f(\xi_1, \xi_2, \ldots, \xi_n) = \eta \quad \text{then} \quad \vdash F(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{\eta}) \text{ in } \mathbf{S} \\ \text{and} \quad \vdash (\forall x_1)(\forall x_2) \ldots (\forall x_n)(!y)\ F(x_1, x_2, \ldots, x_n, y)\ . \end{array}$$

We then say the "F represents f".

Recall that the expression $(!y)\ F(x_1, x_2, \ldots, x_n, y)$ here is short for

$$(\forall y)(\forall y')(\ F(x_1, x_2, \ldots, x_n, y)\ \wedge\ F(x_1, x_2, \ldots, x_n, y') \Rightarrow\ y = y'\ )\ .$$

B.2 Remark

If the function $f : \mathbb{N}^n \to \mathbb{N}$ is representable, then the $(n+1)$-ary relation given by $f(\xi_1, \xi_2, \ldots, \xi_n) = \eta$ is expressible. The first of the two requirements of Part (i) of the definition above is given explicitly. As for the second, suppose that $f(\xi_1, \xi_2, \ldots, \xi_n) \neq \eta$. Then, writing θ for the true value, that is, defining $\theta = f(\xi_1, \xi_2, \ldots, \xi_n)$, we have $\theta \neq \eta$ and so, from these last two equations,

$$\vdash F(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{\theta}) \qquad \text{and} \qquad \vdash \bar{\theta} \neq \bar{\eta}\ .$$

From this and the last line of the definition above it follows that $\vdash \neg F(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{\eta})$.

However, note that the notion of representability of the function says more than that the relation $f(\xi_1, \xi_2, \ldots, \xi_n) = \eta$ is expressible. The last line of the definition above says that, moreover, this relation is of the kind that defines a function.

B.3 Theorem

(i) If a function $f : \mathbb{N}^n \to \mathbb{N}$ is recursive, then it is representable.

(ii) If an n-ary relation on $\mathbb{N}$ is recursive then it is expressible.

Preliminary remarks

We know that, in a formal proof, if we have two lines of the forms $(\forall x)P(x)$ and $(\forall x)Q(x)$ then it is valid to deduce that $(\forall x)(P(x) \wedge Q(x))$. On the other hand, if we have two lines of the forms $(\exists x)P(x)$ and $(\exists x)Q(x)$ then it is **not** valid to deduce that $(\exists x)(P(x) \wedge Q(x))$.

(Think: $(\exists x)\,(x = 2)$ and $(\exists x)\,(x = 3)$ are both true, but $(\exists x)\,(x = 2 \wedge x = 3)$ is not.) It is however valid to deduce that $(\exists x)(\exists x')(P(x) \wedge Q(x'))$.

When using the choice rule, as we will in the proof below, we have a similar consideration. If we have lines of the forms $(\exists x)P(x)$ and $(\exists x)Q(x)$, it is valid to apply the choice rule to the first line to get $P(x)$, but if we apply the choice rule again to the second line, we must introduce a new choice variable and write, say, $Q(x')$.

Proof. (i) Suppose that the function $f : \mathbb{N}^n \to \mathbb{N}$ is recursive. The proof that f is representable is by induction over the construction of f as a recursive function, as described in Corollary **8**.C.6.

► 8.C.6

The successor function $f(\xi) = \xi + 1$ is represented by

$$F(x, y) \;:\; y = x^+$$

The projection function $\pi_{n,i}(\xi_1, \xi_2, \ldots, \xi_n) = \xi_i$ is represented by

$$\Pi_{n,i}(x_1, x_2, \ldots, x_n, y) \;:\; y = x_i \;\wedge\; x_1 = x_1 \;\wedge\; x_2 = x_2 \;\wedge\; \ldots \;\wedge\; x_n = x_n$$

(The apparently redundant parts of this expression are required to satisfy the stipulation in the definition that there be exactly $n + 1$ variables.)

Addition and multiplication are represented by

$$A(x_1, x_2, y) \;:\; x_1 + x_2 = y \qquad \text{and} \qquad M(x_1, x_2, y) \;:\; x_1 x_2 = y\ .$$

Natural subtraction is represented by

$$S(s_1, s_2, y) \;:\; (x_1 < x_2 \;\wedge\; y = 0) \;\vee\; (x_1 = x_2 + y)\,.$$

(The assertions just made require some nit-picking proof.)

Now suppose that $f : \mathbb{N}^n \to \mathbb{N}$ is obtained by substitution, $f = h \circ (g_1, g_2, \ldots, g_m)$, and that h and $g_1, g_2, \ldots, g_m$ are all representable. Then we have expressions $H(z_1, z_2, \ldots, z_m, y)$ and $G(x_1, x_2, \ldots, x_n, z_i)$ for $i = 1, 2, \ldots, m$ such that, for any $\zeta_1, \zeta_2, \ldots, \zeta_m$ and η,

$$\begin{aligned} &\text{If} \quad h(\zeta_1, \zeta_2, \ldots, \zeta_m) = \eta \quad \text{then} \quad \vdash H(\bar\zeta_1, \bar\zeta_2, \ldots, \bar\zeta_m, \bar\eta) \\ \text{and} \quad &\vdash (\forall z_1)(\forall z_2)\ldots(\forall z_m)(!y)\; H(z_1, z_2, \ldots, z_m, y) \end{aligned}$$

and, for any $\xi_1, \xi_2, \ldots, \xi_n, \zeta$ and $i = 1, 2, \ldots, m$,

$$\begin{aligned} &\text{If} \quad g_i(\xi_1, \xi_2, \ldots, \xi_n) = \zeta \quad \text{then} \quad \vdash G_i(\bar\xi_1, \bar\xi_2, \ldots, \bar\xi_n, \bar\zeta) \\ \text{and} \quad &\vdash (\forall x_1)(\forall x_2)\ldots(\forall x_n)(!z)\; G_i(x_1, x_2, \ldots, x_n, z)\,. \end{aligned}$$

We show that f is represented by the (obvious?) expression $F(x_1, x_2, \ldots, x_n, y)$, defined to be

$$\begin{aligned}(\exists z_1)(\exists z_2)\ldots(\exists z_m)\big(\ & H(z_1, z_2, \ldots, z_m, y)\\ & \wedge G_1(x_1, x_2, \ldots, x_n, z_1)\\ & \wedge G_2(x_1, x_2, \ldots, x_n, z_2\\ & \qquad\vdots\\ & \wedge G_m(x_1, x_2, \ldots, x_n, z_m)\ \big)\end{aligned}$$

Suppose that $f \circ (\xi_1, \xi_2, \ldots, \xi_n) = \eta$. Define $\zeta_1, \zeta_2, \ldots, \zeta_m$ by

$$\zeta_i = g_i(\xi_1, \xi_2, \ldots, \xi_n) \qquad \text{for } i = 1, 2, \ldots, m.$$

Then $h(\zeta_1, \zeta_2, \ldots, \zeta_m) = \eta$, and so we have

$$\vdash\ G_i(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{\zeta}_i) \quad \text{for each } i \qquad \text{and} \qquad \vdash H(\bar{\zeta}_1, \bar{\zeta}_2, \ldots, \bar{\zeta}_m, \bar{\eta})\ .$$

Therefore

$$\begin{aligned}\vdash\ & H(\bar{\zeta}_1, \bar{\zeta}_2, \ldots, \bar{\zeta}_m, \bar{\eta})\\ & \wedge G_1(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{\zeta}_1)\\ & \wedge G_2(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{\zeta}_2)\\ & \qquad\vdots\\ & \wedge G_m(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{\zeta}_m)\ .\end{aligned}$$

and so

$$\begin{aligned}\vdash\ (\exists z_1)(\exists z_2)\ldots(\exists z_m)\ \big(& H(z_1, z_2, \ldots, z_m, \bar{\eta})\\ & \wedge G_1(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, z_1)\\ & \wedge G_2(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, z_2)\\ & \qquad\vdots\\ & \wedge G_m(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, z_m)\big)\ .\end{aligned}$$

and this is $\vdash F(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{\eta})$, as required.

We must now show that

$$\vdash\ (\forall x_1)(\forall x_2)\ldots(\forall x_n)(\forall y)(\forall y')(\ F(x_1, x_2, \ldots, x_n, y) \wedge F(x_1, x_2, \ldots, x_n, y')\ \Rightarrow\ y = y'\ )\ .$$

Using Universal Generalisation, it is enough to show that

$$\vdash\ F(x_1, x_2, \ldots, x_n, y) \wedge F(x_1, x_2, \ldots, x_n, y')\ \Rightarrow\ y = y'\ .$$

and then using the Deduction Theorem, it is enough to show that

$$F(x_1, x_2, \ldots, x_n, y)\ ,\ F(x_1, x_2, \ldots, x_n, y') \quad \vdash\ \ y = y'\ .$$

The proof goes as follows: the two hypotheses are

$$\begin{aligned}(\exists z_1)(\exists z_2)\ldots(\exists z_m)\big(\ & H(z_1,z_2,\ldots,z_m,y)\\ & \wedge G_1(x_1,x_2,\ldots,x_n,z_1)\\ & \wedge G_2(x_1,x_2,\ldots,x_n,z_2)\\ & \quad\vdots\\ & \wedge G_m(x_1,x_2,\ldots,x_n,z_m)\ \big)\end{aligned}$$

and

$$\begin{aligned}(\exists z_1)(\exists z_2)\ldots(\exists z_m)\big(\ & H(z_1,z_2,\ldots,z_m,y')\\ & \wedge G_1(x_1,x_2,\ldots,x_n,z_1)\\ & \wedge G_2(x_1,x_2,\ldots,x_n,z_2)\\ & \quad\vdots\\ & \wedge G_m(x_1,x_2,\ldots,x_n,z_m)\ \big)\,.\end{aligned}$$

Using the choice rule on these (and this is where the preliminary remark at the beginning of the proof is relevant), we have

$$\begin{aligned}&H(z_1,z_2,\ldots,z_m,y) && \text{(a)}\\ &G_1(x_1,x_2,\ldots,x_n,z_1) && \text{(a1)}\\ &G_2(x_1,x_2,\ldots,x_n,z_2) && \text{(a2)}\\ &\quad\vdots && (\vdots)\\ &G_m(x_1,x_2,\ldots,x_n,z_m) && \text{(am)}\\ &H(z'_1,z'_2,\ldots,z'_m,y) && \text{(b)}\\ &G_1(x_1,x_2,\ldots,x_n,z'_1) && \text{(b1)}\\ &G_2(x_1,x_2,\ldots,x_n,z'_2) && \text{(b2)}\\ &\quad\vdots && (\vdots)\\ &G_m(x_1,x_2,\ldots,x_n,z'_m) && \text{(bm)}\end{aligned}$$

(as separate lines in our proof). Now, from Lines (a1) and (b1), we have $z'_1 = z_1$ and from the other similar pairs of lines, $z'_2 = z_2\,,\ldots,\ z'_m = z_m$. Then substitution of equals in Line (b) gives $H(z_1,z_2,\ldots,z_m,y')$ which, with Line (a) gives $y' = y$, as required.

Now suppose that f is obtained by minimalisation of a regular function,

$$f(\xi_1,\xi_2,\ldots,\xi_n) = \min_{\eta}\{\ g(\eta,\xi_1,\xi_2,\ldots,\xi_n) = 0\ \}$$

where g is regular and representable, so there is an expression $G(y,x_1,x_2,\ldots,x_n,z)$ such that

$$\text{if}\quad g(\eta,\xi_1,\xi_2,\ldots,\xi_n) = \zeta\quad\text{then}\quad \vdash G(\bar\eta,\bar\xi_1,\bar\xi_2,\ldots,\bar\xi_n,\bar\zeta)$$

and

$$\vdash \quad (\forall y)(\forall x_1)(\forall x_2)\dots(\forall x_n)\ (\ G(y,x_1,x_2,\dots,x_n,z)\ \wedge\ G(y,x_1,x_2,\dots,x_n,z')\ \Rightarrow\ z=z'\ )\,.$$

I will show that f is represented by the (again obvious?) expression $F(x_1,x_2,\dots,x_n,y)$ defined to be

$$G(y,x_1,x_2,\dots,x_n,\bar{0})\ \wedge\ (\forall u<y)\neg G(u,x_1,x_2,\dots,x_n,\bar{0})$$

Firstly, suppose that $f(\xi_1,\xi_2,\dots,\xi_n)=\eta$. Then $g(\eta,\xi_1,\xi_2,\dots,\xi_n)=0$ and $g(\nu,\xi_1,\xi_2,\dots,\xi_n)\neq 0$ for all $\nu<\eta$. We therefore have

$$\vdash\quad G(\bar{\eta},\bar{\xi}_1,\bar{\xi}_2,\dots,\bar{\xi}_n,\bar{0})\,; \tag{9.1}$$

and, for any $\nu<\eta$, using Remark B.2 above, we have ▶ B.2

$$\vdash\quad \neg G(\bar{\nu},\bar{\xi}_1,\bar{\xi}_2,\dots,\bar{\xi}_n,\bar{0})\,.$$

Now this is true for every $\nu<\eta$, so we have

$$\vdash\quad \neg G(\bar{0},\bar{\xi}_1,\bar{\xi}_2,\dots,\bar{\xi}_n,\bar{0})\ \wedge\ \neg G(\bar{1},\bar{\xi}_1,\bar{\xi}_2,\dots,\bar{\xi}_n,\bar{0})\ \wedge\ \dots\ \wedge\ \neg G(\overline{\eta-1},\bar{\xi}_1,\bar{\xi}_2,\dots,\bar{\xi}_n,\bar{0})$$

which, using **4**.B.19, gives ▶ 4.B.19

$$\vdash\quad (\forall u<\bar{\eta})\neg G(u,\bar{\xi}_1,\bar{\xi}_2,\dots,\bar{\xi}_n,\bar{0})\,. \tag{9.2}$$

Now equations 9.1 and 9.2 give $F(\bar{\xi}_1,\bar{\xi}_2,\dots,\bar{\xi}_n,\bar{\eta})$, as required. (That was why we took all the trouble to prove **4**.B.19 back in Chapter 4.) ▶ 4.B.19

Finally we must prove that

$$\vdash\quad (\forall x_1)(\forall x_2)\dots(\forall x_n)(\forall y)(\forall y')\left(F(x_1,x_2,\dots,x_n,y)\ \wedge\ F(x_1,x_2,\dots,x_n,y')\ \Rightarrow\ y=y'\right).$$

Using Universal Generalisation and the Deduction Theorem as before, it is enough to prove that

$$F(x_1,x_2,\dots,x_n,y)\ \wedge\ F(x_1,x_2,\dots,x_n,y')\vdash y=y'\,.$$

The hypotheses are

$$G(y,x_1,x_2,\dots,x_n,\bar{0})\ \wedge\ (\forall u<y)\neg G(u,x_1,x_2,\dots,x_n,\bar{0})$$
$$\text{and}\qquad G(y',x_1,x_2,\dots,x_n,\bar{0})\ \wedge\ (\forall u<y')\neg G(u,x_1,x_2,\dots,x_n,\bar{0})$$

We also have the theorem (**4**.B.11) ▶ 4.B.11

$$\vdash\quad (\forall x)(\forall y)(\quad (x<y)\ \vee\ (x=y)\ \vee\ (y<x)\quad)$$

With the hypotheses, each of $y<y'$ and $y'<y$ easily leads to a contradiction. Hence $\vdash y'=y$ as required.

(ii) Suppose that the n-ary relation r is recursive: we want to show that it is expressible. By definition of recursiveness of a relation, its characteristic function

$$c(\xi_1,\xi_2,\dots,\xi_n)\ =\ \begin{cases}1 & \text{if } r(\xi_1,\xi_2,\dots,\xi_n) \text{ is true,}\\ 0 & \text{if } r(\xi_1,\xi_2,\dots,\xi_n) \text{ is false.}\end{cases}$$

is recursive and so, as just proved above, is representable. Therefore there is an expression $C(x_1, x_2, \ldots, x_n, y)$ which represents it. We will show that r is expressed by the (once again obvious?) expression $C(x_1, x_2, \ldots, x_n, \bar{1})$.

If $r(\xi_1, \xi_2, \ldots, \xi_n)$ is true, then $c(\xi_1, \xi_2, \ldots, \xi_n) = 1$ and so $\vdash C(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{1})$. On the other hand, if $r(\xi_1, \xi_2, \ldots, \xi_n)$ is false, then $c(\xi_1, \xi_2, \ldots, \xi_n) = 0$ and so $\vdash C(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{0})$. From the definition of C we have

$$C(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{0}) \ , \ C(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{1}) \ \vdash \ \bar{0} = \bar{1}$$

and so $\vdash \ C(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{0}) \ \Rightarrow \ (C(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{1}) \ \Rightarrow \ \bar{0} = \bar{1} \ )$

and then by MP $\vdash \ C(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{1}) \ \Rightarrow \ \bar{0} = \bar{1}$.

But of course we also have $\vdash \bar{0} \neq \bar{1}$ and so $\vdash \neg C(\bar{\xi}_1, \bar{\xi}_2, \ldots, \bar{\xi}_n, \bar{1})$, as required. ■

C Gödel's theorem

Recall that in Chapter 4 (Theorems **4**.B.16 and **4**.B.19) the following were shown to be theorems of **S**:

▶ **4**.B.16
▶ **4**.B.19

(i) $\vdash \quad (x = \bar{0}) \ \vee \ (x = \bar{1}) \ \vee \ \ldots \ \vee \ (x = \bar{\nu}) \quad \Leftrightarrow \quad (x \leq \bar{\nu})$

(ii) $\vdash \quad P(\bar{0}) \ \wedge \ P(\bar{1}) \ \wedge \ \ldots \ \wedge \ P(\bar{\nu}) \quad \Leftrightarrow \quad (\forall x \leq \bar{\nu}) P(x)$

These will be used in the proof of Gödel's Theorem below.

We require a slight modification of Lemma A.11:

▶ A.11

C.1 Lemma

The relations

$W(\xi, \eta)$ defined "ξ is the Gödel number of an expression $P(x)$ with exactly one free variable x and η is the Gödel number of a proof of $P(\bar{\xi})$"

and

$V(\xi, \eta)$ defined "ξ is the Gödel number of an expression $P(x)$ with exactly one free variable x and η is the Gödel number of a proof of $\neg P(\bar{\xi})$"

are recursive.

Informal proof. As for Lemma A.11, with a few more obviously algorithmic checks.

▶ A.11

C.2 Gödel's Incompleteness Theorem

If **S** is consistent then it is not complete.

Proof. First, lots of definitions.

In Lemma C.1 above we observed that relation $W(\xi, \eta)$ defined there is recursive. It is therefore expressible, so there is an expression $\hat{W}(x, y)$ whch expresses it, that is,

▶ C.1

$$\text{if } W(\xi,\eta) \text{ is true} \quad \text{then} \quad \vdash \hat{W}(\bar{\xi}, \bar{\eta}) \qquad (\text{–1A})$$

$$\text{and} \qquad \text{if } W(\xi,\eta) \text{ is false} \quad \text{then} \quad \vdash \neg\hat{W}(\bar{\xi}, \bar{\eta}) \qquad (\text{–1B})$$

Define $W^*(x)$ to be the expression $(\forall y)\neg\hat{W}(x, y)$.

Let μ be the Gödel number of $W^*(x)$ and W^{**} be the expression $W^*(\bar{\mu})$.

In the same way the relation $V(\xi, \eta)$ defined above is recursive. It is therefore expressible, and we can define:

$\hat{V}(x, y)$ expresses $V(\xi, \eta)$, that is,

$$\text{if } V(\xi,\eta) \text{ is true} \quad \text{then} \quad \vdash \hat{V}(\bar{\xi}, \bar{\eta}) \qquad (\text{–2A})$$

$$\text{and} \qquad \text{if } V(\xi,\eta) \text{ is false} \quad \text{then} \quad \vdash \neg\hat{V}(\bar{\xi}, \bar{\eta}) \qquad (\text{–2B})$$

Define $V^*(x)$ to be the expression $(\forall y)\big(\hat{W}(x,y) \;\Rightarrow\; (\exists z \le y)\hat{V}(x,z)\big)$.

Let ν be the Gödel number of $V^*(x)$ and V^{**} be the expression $V^*(\bar{\nu})$.

Note that then

$W(\nu,\eta)$ is true if and only if η is the Gödel number of a proof of V^{**}, (–3A)

$V(\nu,\eta)$ is true if and only if η is the Gödel number of a proof of $\neg V^{**}$. (–3B)

We will prove the theorem by showing that, if **S** is consistent, then neither V^{**} nor $\neg V^{**}$ is proveable in **S**. From now on then, assume that **S** is consistent.

i Assume $\vdash V^{**}$, that is,

$$\vdash \quad (\forall y)\big(\; \hat{W}(\bar{\nu},y) \;\Rightarrow\; (\exists z \le y)\hat{V}(\bar{\nu},z)\;\big). \tag{–4}$$

Let α be the Gödel number of a proof of this. Then, by (–3A), $W(\nu,\alpha)$ is true. Then, by (–1A),

$$\vdash \quad \hat{W}(\bar{\nu},\bar{\alpha})\,. \tag{–5}$$

From (–2) and (–3) we have

$$\vdash \quad (\exists z \le \bar{\alpha})\hat{V}(\bar{\nu},z)\,. \tag{–6}$$

We have assumed that **S** is consistent and that $\vdash V^{**}$. It follows that there is no proof of $\neg V^{**}$ in **S**. From (–3B) then

$$V(\nu,\eta) \quad \text{is false for all } \eta \in \mathbb{N}$$

and then by (–2B)

$$\vdash \quad \neg\hat{V}(\bar{\nu},\bar{\eta}) \qquad \text{for all } \eta \in \mathbb{N}\,.$$

In particular,

$$\vdash \neg\hat{V}(\bar{\nu},\bar{0}) \quad , \quad \vdash \neg\hat{V}(\bar{\nu},\bar{1}) \quad , \quad \ldots \quad , \quad \vdash \neg\hat{V}(\bar{\nu},\bar{\alpha})\,.$$

From this

$$\vdash \quad \neg\hat{V}(\bar{\nu},\bar{0}) \;\wedge\; \neg\hat{V}(\bar{\nu},\bar{0}) \;\wedge\; \ldots \;\wedge\; \neg\hat{V}(\bar{\nu},\bar{\alpha})$$

and so, by the theorem recalled at the start of this section,

$$\vdash \quad (\forall z \le \bar{\alpha})\neg\hat{V}(\bar{\nu},z)\,,$$

in other words,

$$\vdash \quad \neg(\exists z \le \bar{\alpha})\hat{V}(\bar{\nu},z)\,,$$

which, with (–6) above, contradicts consistency.

ii Now assume $\vdash \neg V^{**}$, that is,

$$\vdash \quad \neg(\forall y)\big(\; \hat{W}(\bar{\nu},y) \;\Rightarrow\; (\exists z \le y)\hat{V}(\bar{\nu},z)\;\big)\,. \tag{–7}$$

Let β be the Gödel number of a proof of this. Then, by (–3B), $V(\nu,\beta)$ is true. Then, by (–2A),

$$\vdash \quad \hat{V}(\bar{\nu},\bar{\beta})\,. \tag{–8}$$

We are now assuming that **S** is consistent and that $\vdash \neg V^{**}$. It follows that there is no proof of V^{**} in **S**. From (–3A) then,

$$W(\nu,\eta) \quad \text{is false for all } \eta \in \mathbb{N}$$

and then by (–1B)

$$\vdash \quad \neg\hat{W}(\bar{\nu},\bar{\eta}) \qquad \text{for all } \eta \in \mathbb{N}\,.$$

In particular,

$$\vdash \neg\hat{W}(\bar{\nu},\bar{0}) \quad , \quad \vdash \neg\hat{W}(\bar{\nu},\bar{1}) \quad , \quad \ldots \quad , \quad \vdash \neg\hat{W}(\bar{\nu},\bar{\beta})\,.$$

From this, as before,

$$\vdash \quad (\forall y \le \bar{\beta})\neg\hat{W}(\bar{\nu},\bar{\beta})\,. \tag{–9}$$

Some straightforward logical manipulations now. From (–8), trivially,

$$\vdash \quad \bar{\beta} \le y \quad \Rightarrow \quad (\bar{\beta} \le y) \ \wedge \ \hat{V}(\bar{\nu},\bar{\beta})$$

so $$\vdash \quad \bar{\beta} \le y \quad \Rightarrow \quad (\exists z)\big(\ (z \le y) \ \wedge \ \hat{V}(\bar{\nu},z)\ \big)$$

which is $$\vdash \quad \bar{\beta} \le y \quad \Rightarrow \quad (\exists z \le y)\hat{V}(\bar{\nu},z)\ .$$

But also $$\vdash \quad (y \le \bar{\beta}) \ \vee \ (\bar{\beta} \le y)$$

and so $$\vdash \quad (y \le \bar{\beta}) \ \vee \ (\exists z \le y)\hat{V}(\bar{\nu},z)\ .$$

From (–9) $$\vdash \quad \neg\hat{W}(\bar{\nu},y) \ \vee \ (\exists z \le y)\hat{V}(\bar{\nu},z)$$

which is $$\vdash \quad \hat{W}(\bar{\nu},y) \ \Rightarrow \ (\exists z \le y)\hat{V}(\bar{\nu},z)\ .$$

Now, by UG, $\vdash \quad (\forall y)\ \big(\ \hat{W}(\bar{\nu},y) \ \Rightarrow \ (\exists z \le y)\hat{V}(\bar{\nu},z)\ \big)$ which, with (–7), contradicts consistency again. ■

C.3 A fixed-point lemma

Let $P(x)$ be an expression with one free variable x. Then there is a sentence Q with Gödel number ψ such that $\vdash Q \Leftrightarrow P(\bar{\psi})$.

Proof. Let f be the function $\mathbb{N} \to \mathbb{N}$ defined thus: $f(\xi)$ is the Gödel number of the statement obtained by finding the expression with Gödel number ξ and replacing all free variables in it by $\bar{\xi}$. This is clearly recursive, so it is expressed by an expression of exactly two free variables, $F(x,y)$ say:

$$\text{if} \quad f(\xi) = \eta \quad \text{then} \quad \vdash F(\bar{\xi},\bar{\eta})$$

and

$$\vdash \quad (\forall x)(!y)F(x,y)\,. \tag{–1}$$

Given the expression $P(x)$, let $R(x)$ be the expression

$$R(x) \quad \ominus \quad (\exists u)(\ P(u) \wedge F(x,u)\ )$$

and suppose that $R(x)$ has Gödel number ρ. Let Q be $R(\rho)$ and suppose that Q has Gödel number ψ. From this and the definition of f, we see that $\psi = f(\rho)$ and therefore that

$$\vdash \; F(\bar{\rho}, \bar{\psi})\,. \tag{–2}$$

From the definition of Q we also see that it is in fact the expression $(\exists u)(\; P(u) \wedge F(\bar{\rho}, u)\;)$. From this, (–1) and (–2) it follows (by ordinary logic) that

$$\vdash \; Q \;\Leftrightarrow\; P(\bar{\psi})$$

as required. ■

C.4 Church's Theorem

If **S** is consistent then it is not decidable.

In other words, the set $\{\,\nu \;:\; \nu \in \mathbb{N}$ and ν is the Gödel number of a theorem in **S** $\}$ is not recursive.

Note Since **S** is any first order theory which contains Peano Arithmetic, this theorem says that no such theory can be both consistent and decidable.

Proof. We suppose that this set, T say, is recursive and prove that then **S** is not consistent. So we are supposing that the unary relation r on $\mathbb{N}$ defined

$$r(\xi) \;\Leftrightarrow\; \xi \text{ is the Gödel number of a theorem}$$

is recursive. It is then representable, which means that there is an expression $R(x)$ in **S** with exactly one free variable such that, for all $\xi \in \mathbb{N}$,

$$\text{if } \xi \text{ is the Gödel number of a theorem, then } \vdash R(\bar{\xi}) \tag{–1}$$

$$\text{and if not, then } \vdash \neg R(\bar{\xi})\;. \tag{–2}$$

Applying the fixed-point lemma above to $\neg R(x)$, there is a sentence Q with Gödel number ψ such that

$$\vdash \; Q \;\Leftrightarrow\; \neg R(\bar{\psi})\,. \tag{–3}$$

Suppose now that Q is not a theorem; then ψ is not the Gödel number of a theorem, so $\vdash \neg R(\bar{\psi})$ by (–2) and then $\vdash YQ$ by (–3). But this is a contradiction, so YQ is a theorem, that is $\vdash YQ$. But then, by (–1), $\vdash R(\bar{\psi})$. From (–3) again, $\vdash \neg Q$ and so **S** is inconsistent. ■

C.5 Gödel-Rosser Theorem

If **S** is axiomatisable and consistent then it is not complete.

► A.13

Proof. Because if it were complete, then by A.13 it would be decidable, contradicting Church's Theorem above. ■

This is the famous Gödel theorem, as strengthened by Rosser. It says that Peano's axioms are not sufficient to "answer every question about $\mathbb{N}$", in the sense that **PA** is consistent and complete. But it also says much more than that: it is impossible to achieve this result by adding more axioms, provided those axioms are such that it is possible to decide what is an axiom and what isn't.

A. CONSTRUCTION OF THE BASIC NUMBER SYSTEMS

In this appendix I describe the construction of the most important number systems of mathematics, The Natural Numbers $\mathbb{N}$, The Integers $\mathbb{Z}$, The Rational Numbers $\mathbb{Q}$, The Real Numbers $\mathbb{R}$ and, briefly, the complex numbers $\mathbb{C}$.

Proofs will mostly not be given, however the results are given in such an order that each can be proved quite easily from what has gone before.

A Quotient sets and algebraic operations

The ideas of a quotient set and, more specifically, a quotient algebra will be used in several of the constructions that follow. It will make the ensuing discussions simpler if we describe them in general first.

A.1 Quotient sets

Let $\sim$ be an equivalence relation on a set A (see Section **5**.B.15) and let a be a member of A. Then the *equivalence class* of a is the set of all members of A which are equivalent to a. It is denoted $[a]$, so we have

▶ **5**.B.15

$$[a] \;=\; \{\, x \,:\, x \in A \text{ and } x \sim a \,\}\ .$$

It is easy to check that the equivalence classes form a partition of A, that is,

(i) Each equivalence class is nonempty,

(ii) Distinct equivalence classes are disjoint and

(iii) A is the union of all its equivalence classes.

((ii) and (iii) may be restated: every member of A is a member of exactly one equivalence class.) Another useful property is:

(iv) For any $a, b \in A$, the following three things are equivalent: $a \sim b$, $[a] = [b]$ and $[a] \cap [b] \neq \varnothing$.

Given an equivalence relation $\sim$ on a set A, we define the *quotient set* $A/\!\sim$ to be the set of all equivalence classes under $\sim$:

$$A/\!\sim \;=\; \{\, [a] \,:\, a \in A \,\}\ .$$

The function $a \mapsto [a]$ is called the *natural projection* $A \to A/\!\sim$.

A.2 Algebraic operations

An *algebraic operation* of *arity* n on a set A is simply a function $A^n \to A$. Examples of algebraic operations are addition and multiplication on $\mathbb{N}$ (both binary), and indeed addition and multiplication on the rest of our number systems ($\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$). Negation is a unary operation on all of these systems except for $\mathbb{N}$. Inversion and division are not algebraic operations because they are not everywhere defined (x/y and y^{-1} are undefined when y is zero). Subtraction is a binary algebraic operation on most of our number systems, but not on $\mathbb{N}$.

A.3 Respect

We say that an equivalence relation $\equiv$ on a set A *respects* the algebraic operation θ if (assuming θ is n-ary), for any members $a_1, a_2, \ldots, a_n$ and $a'_1, a'_2, \ldots, a'_n$ of A such that $a_1 \equiv a'_1$, $a_2 \equiv a'_2$, $\ldots$, $a_n \equiv a'_n$,

$$\theta(a_1, a_2, \ldots, a_n) \;\equiv\; \theta(a'_1, a'_2, \ldots, a'_n) \; .$$

An equivalence relation which respects all the algebraic operations we are interested in is usually called a *congruence*.

Observe that any nullary operation is automatically respected by any equivalence relation (the definition above is vacuously satisfied).

The most familiar example of a congruence is ordinary congruence modulo n on the integers, for any fixed integer n. It respects all the ordinary operations of a ring with unity: addition, negation, zero, multiplication and unity (one).

Now let $\equiv$ be an equivalence relation on an algebra which respects the n-ary algebraic operation θ. Then θ can also be defined on the quotient set in a natural way as follows: let $X_1, X_2, \ldots, X_n$ be members of $A/\!\equiv$, that is, equivalence classes. Choose any members $x_1, x_2, \ldots, x_n$ of these classes ($x_1 \in X_1, x_2 \in X_2, \ldots, x_n \in X_n$). Then $\theta(X_1, X_2, \ldots, X_n)$ is defined to be the equivalence class containing $\theta(x_1, x_2, \ldots, x_n)$.

It must of course be checked that this definition makes sense: that the definition just given of $\theta(X_1, X_2, \ldots, X_n)$ is independent of the particular choice of the representatives $x_1, x_2, \ldots, x_n$ of the classes. That this is so follows quite easily from the fact that $\equiv$ respects θ.

Once this has been established we can rewrite the definition in a much more convenient form: given $\equiv$ and θ as above, let $a_1, a_2, \ldots, a_n$ be members of A; then $[a_1], [a_2], \ldots, [a_n]$ are equivalence classes and we want to define $\theta([a_1], [a_2], \ldots, [a_n])$. Well then,

$$\theta([a_1], [a_2], \ldots, [a_n]) \;=\; [\theta(a_1, a_2, \ldots, a_n)] \; .$$

A.4 Laws

A *law* (in the sense in which we will be using the the word here) is an equation involving the defined operations which must be true for all values of the variables involved. Examples are the commutative law

$$(\forall x \in A)(\forall y \in A) \; (\; xy = yx \;)$$

and the associative law

$$(\forall x \in A)(\forall y \in A)(\forall z \in A)\ (\ (xy)z = x(yz)\ )\ .$$

Because of what we are about to do, it is a good idea to make sure that this definition is precisely understood. Firstly, we define *terms* as in Chapter **3**: a term is either a variable symbol $x, y, z \ldots$ or of the form $\theta(t_1, t_2, \ldots, t_n)$, where θ is an n-ary operation and $t_1, t_2, \ldots, t_n$ are (simpler) terms. Next, we define an *equation* as an expression of the form ▶ Ch.3

$$\text{term}_1 \ = \ \text{term}_2$$

where term_1 and term_2 are of course terms. Finally, a law is an expression of the form

$$(\forall x \in A)(\forall y \in A)(\forall z \in A)\ \text{equation}\ ,$$

were $x_1, x_2, \ldots, x_n$ are *all* the variable symbols which appear in the equation.

Note that there are a number of facts which are often called laws which are not in fact laws in the sense just defined. For instance, the "cancellative law of addition" for $\mathbb{Z}$ states:

$$(\forall x \in \mathbb{Z})(\forall y \in \mathbb{Z})(\exists z \in \mathbb{Z})\ (\ x + z = y\ )\ .$$

The existential quantifier here stops this from being a law. In the same way, facts about fields involving division fail to be laws; for example

$$(\forall x \in \mathbb{Q})\ (\ x \neq 0\ \Rightarrow\ x/x = 1\ )\ .$$

The implication here is not allowed.

The point of all this, as far as this appendix goes, is the following very useful theorem.

A.5 Theorem

Let A be an algebra and $\equiv$ be an equivalence relation on A. Then any law of A which involves only operations which are respected by $\equiv$ is also a law of $A/\!\equiv$.

A.6 Orders

Most of the above discussion applies to relations in the same way. We are only interested in order relations here, so I will summarise the relevant facts for orders; the generalisations to arbitrary relations, should you be interested, are obvious.

Recall that a preorder on a set A is a binary relation, $\leq$ say, with the properties:

Reflexivity: For all $a \in A$, $\quad a \leq a$.

Transitivity: For all $a, b, c \in A$, $\quad a \leq b\ \wedge\ b \leq c\ \Rightarrow\ a \leq c$.

A partial order is a preorder with the extra property:

Antisymmetry: For all $a, b \in A$, $\quad a \leq b\ \wedge\ b \leq a\ \Rightarrow\ a = b$.

A full order is a partial order with the extra property:

Dichotomy: For all $a, b \in A$, either $a \leq b$ or $b \leq a$.

We may also speak of a "full preorder", which is a relation which has all of these properties except possibly antisymmetry.

It is easy to check that, if $\leq$ is a preorder on the set A, then the relation $\sim$ defined by

$$x \sim y \quad \Leftrightarrow \quad x \leq y \text{ and } y \leq x$$

is an equivalence relation. We call this the equivalence relation *defined by* (or *corresponding to*) $\leq$.

Now let A be a set and $\leq$ be a preorder on A. Let $\sim$ be an equivalence relation (not necessarily the one defined by $\leq$). We say that $\sim$ *respects* the order relation $\leq$ if, for all $a, b, a', b' \in A$

$$a \sim a' \text{ and } b \sim b' \text{ and } a \leq b \quad \Rightarrow \quad a' \leq b' \ .$$

In this case a corresponding preorder can be defined on the quotient set $A/\sim$ in the obvious way: let X and Y be equivalence classes in A. Choose any members x of X and y of Y. Then define $X \leq Y$ if and only if $x \leq y$.

Having checked, as before, that this definition makes sense, that is, that the definition of $X \leq Y$ is independent of the particular members x and y chosen, we may rewrite the definition in the more convenient form: for all $a, b \in A$, $[a] \leq [b]$ if and only if $a \leq b$.

Now the following facts are easily checked:

(1) If the order $\leq$ was a partial order, a full order or a full preorder on A, then the corresponding order on $A/\sim$ is an order of the same kind.

(2) If the order $\leq$ was a preorder on A and the relation $\sim$ happens to be the equivalence relation defined by $\leq$, then the corresponding order on $A/\sim$ is a partial order.

Note that, combining (1) and (2), if the order $\leq$ was a full preorder and the relation $\sim$ is the equivalence relation defined by $\leq$, then the corresponding order on $A/\sim$ is a full order.

B The Natural Numbers, $\mathbb{N}$

This has already been defined and discussed in Section **5**.D. Here we will introduce its algebraic and order properties. ▶ **5**.D

B.1 Fundamental properties of The Natural Numbers

Addition, multiplication and the order relation are defined on $\mathbb{N}$ as follows:

(N1a) For all $a \in \mathbb{N}$, $a + 0 = a$.

(N1b) For all $a, b \in \mathbb{N}$, $a + b^+ = (a + b)^+$.

(N2a) For all $a \in \mathbb{N}$, $a.0 = 0$.

(N2b) For all $a, b \in \mathbb{N}$, $ab^+ = (ab) + a$.

(N3) For all $a, b \in \mathbb{N}$, $a \leq b \quad \Leftrightarrow \quad (\exists u)(a + u = b)$

Now we can prove the following laws by induction.

(N4) Addition is associative. For all $a, b, c \in \mathbb{N}$, $(a + b) + c = a + (b + c)$.

(N5) Addition is commutative. For all $a, b \in \mathbb{N}$, $a + b = b + a$.

(N6) Multiplication is associative. For all $a, b, c \in \mathbb{N}$, $(ab)c = a(bc)$.

(N7) Multiplication distributes over addition. For all $a, b, c \in \mathbb{N}$, $a(b + c) = ab + ac$.

(N8) Multiplication is commutative. For all $a, b \in \mathbb{N}$, $ab = ba$.

(N9) Addition is cancellative. If $a + x = a + y$ then $x = y$.

(N10) Multiplication is cancellative. If $ax = ay$ and $a \neq 0$ then $x = y$.

The relation $\leq$ is a well-order:

(N11a) For all $a \in \mathbb{N}$, $a \leq a$.

(N11b) For all $a, b, c \in \mathbb{N}$, if $a \leq b$ and $b \leq c$ then $a \leq c$.

(N11c) For all $a, b \in \mathbb{N}$, if $a \leq b$ and $b \leq a$ then $a = b$.

(N11d) For all $a, b \in \mathbb{N}$, $a \leq b$ or $b \leq a$.

(N11w) Every nonempty subset of $\mathbb{N}$ has a least element.

C The Integers, $\mathbb{Z}$

C.1 Preliminary

We are about to construct the Integers $\mathbb{Z}$ from the Natural Numbers $\mathbb{N}$. Let us suppose for a moment that we already have $\mathbb{Z}$ in some sense to look at. How can we specify a member z of $\mathbb{Z}$ using natural numbers? Well we can write any integer $z = m - n$, where m and n are natural numbers, and so we can use the pair $\langle m, n\rangle$ to represent z. Thus, as a first approximation, we might consider defining z to actually *be* the pair $\langle m, n\rangle$, and so construct $\mathbb{Z}$ as the set $\mathbb{N} \times \mathbb{N}$ of pairs of natural numbers.

There is a problem with this however: any given integer z can be represented by a pair of natural numbers in many ways: $z = m - n = m' - n'$. To accommodate this we consider each integer z to actually be the *set* of pairs of natural numbers which represent it. Thus we will define $\mathbb{Z}$ to be a set of appropriately defined subsets of $\mathbb{N} \times \mathbb{N}$, upon which we will define algebraic operations and order.

In order to define which subsets of $\mathbb{N} \times \mathbb{N}$ are going to represent integers, we define a relation between pairs: $\langle m, n\rangle \sim \langle m', n'\rangle$ if they represent the same integer, that is, if $m - n = m' - n'$. This turns out to be an equivalence relation, and so we can define $\mathbb{Z}$ conveniently as the quotient set $(\mathbb{N} \times \mathbb{N})/\sim$.

There is one last small problem to be got out of the way: we need to be able to define the relation $\sim$ in order to construct $\mathbb{Z}$. But before $\mathbb{Z}$ has been constructed, the equation $m - n = m' - n'$ has no meaning (certainly not in $\mathbb{N}$ when $m < n$). Using our hindsight, we observe that this equation is the same as $m + n' = m' + n$ and that this one does make sense in $\mathbb{N}$, so we use this as our definition.

We are going to construct $\mathbb{Z}$ as a quotient set of $\mathbb{N} \times \mathbb{N}$. It will be convenient to take this in two steps, and discuss $\mathbb{N} \times \mathbb{N}$, with some useful structure added, as a sort of half-way-there construction.

C.2 Construction of The Integers, Step 1

Let us write J for the set $\mathbb{N} \times \mathbb{N}$, with some operations defined as follows.

Zero	$0_J = \langle 0, 0\rangle$
Addition	$\langle a, b\rangle + \langle c, d\rangle = \langle a + c, b + d\rangle$
Negation	$-\langle a, b\rangle = \langle b, a\rangle$
One	$1_J = \langle 1, 0\rangle$
Multiplication	$\langle a, b\rangle\langle c, d\rangle = \langle ac + bd, ad + bc\rangle$

And the relations

Equivalence	$\langle a, b\rangle \sim \langle c, d\rangle$	$\Leftrightarrow$	$a + d = b + c$
Preorder	$\langle a, b\rangle \leq \langle c, d\rangle$	$\Leftrightarrow$	$a + d \leq b + c$

J1 We now prove that all the usual algebraic and order laws for $\mathbb{Z}$ hold in J "up to equivalence", as follows:

(i) $(\langle a,b\rangle + \langle c,d\rangle) + \langle e,f\rangle \sim \langle a,b\rangle + (\langle c,d\rangle + \langle e,f\rangle)$

(ii) $\langle a,b\rangle + 0_J \sim \langle a,b\rangle$

(iii) $\langle a,b\rangle + (-\langle a,b\rangle) \sim 0_J$

(iv) $\langle a,b\rangle + \langle c,d\rangle \sim \langle c,d\rangle + \langle a,b\rangle$

(v) $(\langle a,b\rangle\langle c,d\rangle)\langle e,f\rangle \sim \langle a,b\rangle(\langle c,d\rangle\langle e,f\rangle)$

(vi) $\langle a,b\rangle(\langle c,d\rangle + \langle e,f\rangle) \sim \langle a,b\rangle\langle c,d\rangle + \langle a,b\rangle\langle e,f\rangle$

(vii) $\langle a,b\rangle.1_J \sim \langle a,b\rangle$

(viii) $\langle a,b\rangle\langle c,d\rangle \sim \langle c,d\rangle\langle a,b\rangle$

(ix) The relation $\leq$ defined on J is a preorder and its corresponding equivalence relation is $\sim$.

(x) $\langle a,b\rangle \leq \langle c,d\rangle \quad \Rightarrow \quad \langle a,b\rangle + \langle e,f\rangle \leq \langle c,d\rangle + \langle e,f\rangle$

(xi) $\langle a,b\rangle \leq \langle c,d\rangle$ and $0_J \leq \langle e,f\rangle \quad \Rightarrow \quad \langle a,b\rangle\langle e,f\rangle \leq \langle c,d\rangle\langle e,f\rangle$

Now (ix) above tells us that $\sim$ is an equivalence relation. Next we prove that it respects all the operations we have defined on J:

J2 If $\langle a,b\rangle \sim \langle a',b'\rangle$ and $\langle c,d\rangle \sim \langle c',d'\rangle$ then

(i) $\langle a,b\rangle + \langle c,d\rangle \sim \langle a',b'\rangle + \langle c',d'\rangle$

(ii) $-\langle a,b\rangle \sim -\langle a',b'\rangle$

(iii) $\langle a,b\rangle\langle c,d\rangle \sim \langle a',b'\rangle\langle c',d'\rangle$

(iv) $\langle a,b\rangle \leq \langle c,d\rangle \quad \Leftrightarrow \quad \langle a',b'\rangle \leq \langle c',d'\rangle$.

Now the scene is set to construct $\mathbb{Z}$ itself.

C.3 Construction of The Integers, Step 2

We define $\mathbb{Z}$ to be the quotient set: $\mathbb{Z} = (\mathbb{N}\times\mathbb{N})/\!\sim \; = J/\!\sim$.

We use the notation $[a,b]$ for the equivalence class of $\langle a,b\rangle$, so that $[a,b] = [c,d] \;\Leftrightarrow\; a+d = b+c$.

Z1 From (J2) it follows that we can define corresponding operations on $\mathbb{Z}$ by

(o) $0_{\mathbb{Z}} = [0,0] \qquad$ and $\qquad 1_{\mathbb{Z}} = [1,0]$

(i) $[a,b] + [c,d] = [a+c, b+d]$ (= the equivalence class of $\langle a,b\rangle + \langle c,d\rangle$).

(ii) $-[a,b] = [b,a]$ (= the equivalence class of $-\langle a,b\rangle$).

(iii) $[a,b][c,d] = [ac+bd, ad+bc]$ (= the equivalence class of $\langle a,b\rangle\langle c,d\rangle$).

(iv) $[a,b] \le [c,d] \quad \Leftrightarrow \quad a+d \le b+c \quad (\Leftrightarrow \quad \langle a,b\rangle \le \langle c,d\rangle)$.

(The point here is that the results of (J2) mean that the definitions here are independent of the particular members chosen to represent the equivalence classes.)

Z2 Applying (Z1) to (J1) gives all the basic laws for $\mathbb{Z}$.

(i) $([a,b]+[c,d])+[e,f] = [a,b]+([c,d]+[e,f])$

(ii) $[a,b]+0_{\mathbb{Z}} = [a,b]$

(iii) $[a,b]+(-[a,b]) = 0_{\mathbb{Z}}$

(iv) $[a,b]+[c,d] = [c,d]+[a,b]$

(v) $([a,b][c,d])[e,f] = [a,b]([c,d][e,f])$

(vi) $[a,b]([c,d]+[e,f]) = [a,b][c,d]+[a,b][e,f]$

(vii) $[a,b].1_{\mathbb{Z}} = [a,b]$

(viii) $[a,b][c,d] = [c,d][a,b]$

(ix) The relation $\le$ defined on $\mathbb{Z}$ is a full order.

(x) $[a,b] \le [c,d] \quad \Rightarrow \quad [a,b]+[e,f] \le [c,d]+[e,f]$

(xi) $[a,b] \le [c,d]$ and $0_{\mathbb{Z}} \le [e,f] \quad \Rightarrow \quad [a,b][e,f] \le [c,d][e,f]$

Z3 All these terms in (Z2) above are simply arbitrary integers, so the laws look more familiar in more normal notation:

(i) $(x+y)+z = x+(y+z)$

(ii) $x+0 = x$

(iii) $x+(-x) = 0$

(iv) $x+y = y+x$

(v) $(xy)z = x(yz)$

(vi) $x(y+z) = xy+xz$

(vii) $x.1 = x$

(viii) $xy = yx$

(ix) The relation $\le$ defined on $\mathbb{Z}$ is a full order.

(x) $x \le y \quad \Rightarrow \quad x+z \le y+z$

(xi) $x \le y$ and $0 \le z \quad \Rightarrow \quad xz \le yz$.

Z4 Finally, $\mathbb{N}$ is embedded in $\mathbb{Z}$ by the map $n \mapsto [n,0]$. In other words, the function

$\iota : \mathbb{N} \to \mathbb{Z}$ defined by $\iota(n) = [n, 0]$ for all $n \in \mathbb{N}$ is an injection which preserves all the operations and the order. Thus the image of $\mathbb{N}$ in $\mathbb{Z}$ under this mapping is an identical copy of $\mathbb{N}$ in all salient respects. From now on we may, and often do when it suits us, think of $\mathbb{N}$ as this subset of $\mathbb{Z}$.

D The Rationals, $\mathbb{Q}$

D.1 Preliminary

Now we construct The Rationals $\mathbb{Q}$ from The Integers $\mathbb{Z}$. The general method is exactly the same as for the previous construction, with only a few small changes of detail. Using hindsight as before, we know that any rational q can be written as a quotient $\frac{a}{b}$, where a and b are integers and $b \neq 0$. Of course, any particular rational can be written as such a product in many ways, and this gives rise to an equivalence relation on $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$, $\langle a, b\rangle \sim \langle a', b'\rangle$ if and only if $\frac{a}{b} = \frac{a'}{b'}$. This can be defined within $\mathbb{Z}$ by $ab' = a'b$.

We are going to construct $\mathbb{Q}$ as a quotient set of $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$, the set of all pairs $\langle a, b\rangle$ of integers such that $b \neq 0$. As before, it will be convenient to take the construction in two steps, and first discuss this product set, with some useful structure added, as a sort of half-way-there construction.

D.2 Construction of The Rationals, Step 1

Let us write K for the set $\mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$, with some operations defined as follows.

Zero	$0_K = \langle 0, 1\rangle$
Addition	$\langle a, b\rangle + \langle c, d\rangle = \langle ad + bc, bd\rangle$
Negation	$-\langle a, b\rangle = \langle -a, b\rangle$
One	$1_K = \langle 1, 1\rangle$
Multiplication	$\langle a, b\rangle\langle c, d\rangle = \langle ac, bd\rangle$
Inversion	If $a \neq 0$, then $\langle a, b\rangle^{-1} = \langle b, a\rangle$

And the relations

Equivalence	$\langle a, b\rangle \sim \langle c, d\rangle \quad \Leftrightarrow \quad ad = bc$
Preorder	$\langle a, b\rangle \leq \langle c, d\rangle \quad \Leftrightarrow \quad (bd > 0$ and $ad \leq bc)$ or $(bd < 0$ and $ad \geq bc)$.

K1 We prove that all the usual algebraic and order laws for $\mathbb{Q}$ hold in K "up to equivalence", as follows:

(i) $(\langle a, b\rangle + \langle c, d\rangle) + \langle e, f\rangle \sim \langle a, b\rangle + (\langle c, d\rangle + \langle e, f\rangle)$

(ii) $\langle a, b\rangle + 0_K \sim \langle a, b\rangle$

(iii) $\langle a, b\rangle + (-\langle a, b\rangle) \sim 0_K$

(iv) $\langle a, b\rangle + \langle c, d\rangle \sim \langle c, d\rangle + \langle a, b\rangle$

(v) $(\langle a, b\rangle\langle c, d\rangle)\langle e, f\rangle \sim \langle a, b\rangle(\langle c, d\rangle\langle e, f\rangle)$

(vi) $\langle a, b\rangle(\langle c, d\rangle + \langle e, f\rangle) \sim \langle a, b\rangle\langle c, d\rangle + \langle a, b\rangle\langle e, f\rangle$

(vii) $\langle a,b\rangle.1_K \sim \langle a,b\rangle$ **(viii)** $\langle a,b\rangle\langle c,d\rangle \sim \langle c,d\rangle\langle a,b\rangle$

(ix) If $\langle a,b\rangle \not\sim 0_K$, then $\langle a,b\rangle\langle a,b\rangle^{-1} \sim 1_K$

(x) The relation $\leq$ defined on K is a preorder and its corresponding equivalence relation is $\sim$.

(xi) $\langle a,b\rangle \leq \langle c,d\rangle \quad \Rightarrow \quad \langle a,b\rangle + \langle e,f\rangle \leq \langle c,d\rangle + \langle e,f\rangle$

(xii) $\langle a,b\rangle \leq \langle c,d\rangle$ and $0_K \leq \langle e,f\rangle \quad \Rightarrow \quad \langle a,b\rangle\langle e,f\rangle \leq \langle c,d\rangle\langle e,f\rangle$

Now (x) above tells us that $\sim$ is an equivalence relation. Next we prove that it respects all the operations we have defined on H:

K2 If $\langle a,b\rangle \sim \langle a',b'\rangle$ and $\langle c,d\rangle \sim \langle c',d'\rangle$ then

(i) $\langle a,b\rangle + \langle c,d\rangle \sim \langle a',b'\rangle + \langle c',d'\rangle$

(ii) $-\langle a,b\rangle \sim -\langle a',b'\rangle$

(iii) $\langle a,b\rangle\langle c,d\rangle \sim \langle a',b'\rangle\langle c',d'\rangle$

(iv) $a \neq 0$ if and only if $a' \neq 0$ and then $\langle a,b\rangle^{-1} \sim \langle a',b'\rangle^{-1}$

(v) $\langle a,b\rangle \leq \langle c,d\rangle \quad \Leftrightarrow \quad \langle a',b'\rangle \leq \langle c',d'\rangle$.

Now we are ready to construct $\mathbb{Q}$ itself.

D.3 Construction of The Rationals, Step 2

We define $\mathbb{Q}$ to be the quotient set: $\mathbb{Q} = (\mathbb{Z} \times (\mathbb{Z} \setminus \{0\}))/\sim \; = K/\sim$.

We use the notation $\frac{a}{b}$ for the equivalence class of $\langle a,b\rangle$, so that $\frac{a}{b} = \frac{c}{d} \;\Leftrightarrow\; ad = bc$.

Q1 From (K3) it follows that we can define corresponding operations on $\mathbb{Q}$ by

(o) $0_{\mathbb{Q}} = \frac{0}{1}$ and $1_{\mathbb{Q}} = \frac{1}{1}$

(i) $\frac{a}{b} + \frac{c}{d} = \frac{ad+bc}{bd}$ (= the equivalence class of $\langle a,b\rangle + \langle c,d\rangle$).

(ii) $-\frac{a}{b} = \frac{-a}{b}$ (= the equivalence class of $-\langle a,b\rangle$).

(iii) $\frac{a}{b}\,\frac{c}{d} = \frac{ac}{bd}$ (= the equivalence class of $\langle a,b\rangle\langle c,d\rangle$).

(iv) If $\frac{a}{b} \neq 0_{\mathbb{Q}}$, then $(\frac{a}{b})^{-1} = \frac{b}{a}$ (= the equivalence class of $\langle a,b\rangle^{-1}$).

(v) $\frac{a}{b} \leq \frac{c}{d} \quad \Leftrightarrow \quad (bd > 0$ and $ad \leq bc)$ or $(bd < 0$ and $ad \geq bc) \quad (\Leftrightarrow \quad \langle a,b\rangle \leq \langle c,d\rangle)$.

(The point here is that the results of (K2) mean that the definitions here are independent of the particular members chosen to represent the equivalence classes.)

Q2 Applying (Q1) to (K1) gives all the basic laws for $\mathbb{Q}$.

(i) $(\frac{a}{b} + \frac{c}{d}) + \frac{e}{f} = \frac{a}{b} + (\frac{c}{d} + \frac{e}{f})$

(ii) $\frac{a}{b} + 0 = \frac{a}{b}$ **(iii)** $\frac{a}{b} + (-\frac{a}{b}) = 0$

(iv) $\frac{a}{b} + \frac{c}{d} = \frac{c}{d} + \frac{a}{b}$

(v) $(\frac{a}{b}\ \frac{c}{d})\frac{e}{f} = \frac{a}{b}(\frac{c}{d}\ \frac{e}{f})$

(vi) $\frac{a}{b}(\frac{c}{d} + \frac{e}{f}) = \frac{a}{b}\ \frac{c}{d} + \frac{a}{b}\ \frac{e}{f}$

(vii) $\frac{a}{b}.1 = \frac{a}{b}$

(viii) $\frac{a}{b}\frac{c}{d} = \frac{c}{d}\frac{a}{b}$

(ix) If $\frac{a}{b} \neq 0$ then $\frac{a}{b}(\frac{a}{b})^{-1} = 1$

(x) The relation $\leq$ defined on $\mathbb{Q}$ is a full order.

(xi) $\frac{a}{b} \leq \frac{c}{d} \Rightarrow \frac{a}{b} + \frac{e}{f} \leq \frac{c}{d} + \frac{e}{f}$

(xii) $\frac{a}{b} \leq \frac{c}{d}$ and $0 \leq \frac{e}{f} \Rightarrow \frac{a}{b}\frac{e}{f} \leq \frac{c}{d}\frac{e}{f}$

Q3 All these terms in (Q2) above are simply arbitrary rationals, so the laws look more familiar in more normal notation:

(i) $(x + y) + z = x + (y + z)$

(ii) $x + 0 = x$

(iii) $x + (-x) = 0$

(iv) $x + y = y + x$

(v) $(xy)z = x(yz)$

(vi) $x(y + z) = xy + xz$

(vii) $x.1 = x$

(viii) $xy = yx$

(ix) If $x \neq 0$ then $x(x)^{-1} = 1$

(x) The relation $\leq$ defined on $\mathbb{Q}$ is a full order.

(xi) $x \leq y \Rightarrow x + z \leq y + z$

(xii) $x \leq y$ and $0 \leq z \Rightarrow xz \leq yz$

Q4 Finally, $\mathbb{Z}$ is embedded in $\mathbb{Q}$ by the map $n \mapsto \frac{n}{1}$. In other words, the function $\iota : \mathbb{Z} \to \mathbb{Q}$ defined by $\iota(n) = \frac{n}{1}$ for all $n \in \mathbb{Z}$ is an injection which preserves all the operations and the order. Thus the image of $\mathbb{Z}$ in $\mathbb{Q}$ under this mapping is an identical copy of $\mathbb{Z}$ in all salient respects. From now on we may, and often do when it suits us, think of $\mathbb{Z}$ as this subset of $\mathbb{Q}$.

E The Reals, $\mathbb{R}$

E.1 Preliminary

Once again we use hindsight and consider what we know about the reals to decide how to construct them from the rationals. There are two standard ways of doing this and I will describe them both here; the first is the "completion" method, in which real numbers are obtained as limits of convergent sequences of rational numbers, and the second is the method of Dedekind cuts.

First we set out to define a real in terms of rational numbers is as the limit of a convergent sequence. The usual definition of convergence, that the sequence $\langle a_0, a_1, a_2, \ldots\rangle$ converges to the real x if

$$(\forall \text{ real } \epsilon > 0)(\exists n \in \mathbb{N})(\forall i \in \mathbb{N})(i \geq n \quad \Rightarrow \quad |a_i - x| < \epsilon),$$

won't do, because we need to be able to identify a convergent sequence of rationals *before* the reals have been constructed — that is, we need to be able to recognise, before the reals have been constructed, those sequences of rationals which are going to converge after the reals have been constructed. The definition above mentions reals in two places: firstly in $(\forall \text{ real } \epsilon > 0)$ and secondly in the limit x. The first occurrence is no real problem: $(\forall \text{ rational } \epsilon > 0)$ will do just as well. The second occurrence is more of a problem: the definition is fine if you already have x to work with, but we are here trying to create x out of thin air.

The solution is to use Cauchy sequences. A sequence $\langle a_0, a_1, a_2, \ldots\rangle$ of real numbers is a *Cauchy sequence* if

$$(\forall \text{ real } \epsilon > 0)(\exists n \in \mathbb{N})(\forall i, j \in \mathbb{N})(i, j \geq n \quad \Rightarrow \quad |a_i - a_j| < \epsilon). \tag{1}$$

That is the usual definition, but it is easy to see that replacing "real" by "rational" in $(\forall \text{ real } \epsilon > 0)$ results in an equivalent definition. Also, we know two crucial things about Cauchy sequences: (1) Cauchy sequences are convergent (in the ordinary sense above) and (2) Every real number is the limit of a Cauchy sequence of rationals. Just to make sure there is no confusion, for the rest of this section a *Cauchy sequence* will mean a Cauchy sequence of rational numbers, unless otherwise stated, that is, a sequence $\langle a_0, a_1, a_2, \ldots\rangle$ of rational numbers satisfying

$$(\forall \text{ rational } \epsilon > 0)(\exists n \in \mathbb{N})(\forall i, j \in \mathbb{N})(i, j \geq n \quad \Rightarrow \quad |a_i - a_j| < \epsilon). \tag{2}$$

Just as in our previous constructions, we must observe that each real number is going to be the limit of many Cauchy sequences. This defines an equivalence relation on the Cauchy sequences, but we need a way of defining it *a priori* — without reference to real numbers. This is not too difficult: two Cauchy sequences $\langle a_0, a_1, a_2, \ldots\rangle$ and $\langle b_0, b_1, b_2, \ldots\rangle$ are equivalent in this sense if and only if

$$(\forall \text{ rational } \epsilon > 0)(\exists n \in \mathbb{N})(\forall i \in \mathbb{N})(i \geq n \quad \Rightarrow \quad |a_i - b_i| < \epsilon).$$

Having made these observations, the construction of $\mathbb{R}$ follows the now familiar pattern.

We are going to construct $\mathbb{R}$ as a quotient set of the set of all Cauchy sequences of rationals. As before, it will be convenient to take this in two steps, and discuss the set of all Cauchy sequences of rationals, with some useful structure added, as a half-way-there construction.

E.2 Construction of The Reals by completion, Step 1

Let us write L for the set of all Cauchy sequences of rationals, with operations defined as follows.

Zero	$0_L = \langle 0, 0, 0, \ldots \rangle$
Addition	$\langle a_0, a_1, a_2, \ldots \rangle + \langle b_0, b_1, b_2, \ldots \rangle = \langle a_0 + b_0, a_1 + b_1, a_2 + b_2, \ldots \rangle$
Negation	$-\langle a_0, a_1, a_2, \ldots \rangle = \langle -a_0, -a_1, -a_2, \ldots \rangle$
One	$1_L = \langle 1, 1, 1, \ldots \rangle$
Multiplication	$\langle a_0, a_1, a_2, \ldots \rangle \langle b_0, b_1, b_2, \ldots \rangle = \langle a_0 b_0, a_1 b_1, a_2 b_2, \ldots \rangle$
Inversion	$\langle a_0, a_1, a_2, \ldots \rangle^{-1} = \langle a_0^*, a_1^*, a_2^*, \ldots \rangle$
	(where for each i, $a_i^* = a_i^{-1}$ if $a_i \neq 0$ and $a_i^* = 0$ if $a_i = 0$).

and relations defined

Equivalence	$\langle a_0, a_1, a_2, \ldots \rangle \sim \langle b_0, b_1, b_2, \ldots \rangle \quad \Leftrightarrow$
	$(\forall \text{ rational } \epsilon > 0)(\exists n \in \mathbb{N})(\forall i \in \mathbb{N})(i \geq n \quad \Rightarrow \quad \lvert a_i - b_i \rvert < \epsilon).$
Preorder	$\langle a_0, a_1, a_2, \ldots \rangle \leq \langle b_0, b_1, b_2, \ldots \rangle \quad \Leftrightarrow$
	$(\forall \text{ rational } \epsilon > 0)(\exists n \in \mathbb{N})(\forall i \in \mathbb{N})(i \geq n \quad \Rightarrow \quad b_i - a_i < \epsilon).$

We prove that all of these operations, when applied to Cauchy sequences, yield Cauchy sequences (with the exception of inversion when applied to a sequence which is equivalent to 0_L).

L1 As before, we prove all the basic algebraic and order laws on L "up to equivalence". Let X, Y and Z be any three Cauchy sequences in L. Then

(i) $(X + Y) + Z \sim X + (Y + Z)$

(ii) $X + 0_L \sim X$

(iii) $X + (-X) \sim 0$

(iv) $X + Y \sim Y + X$

(v) $(XY)Z \sim X(YZ)$

(vi) $X(Y + Z) \sim XY + XZ$

(vii) $X.1_L \sim X$

(viii) $XY \sim YX$

(ix) If $X \not\sim 0$ then $X(X)^{-1} \sim 1_L$

(x) The relation $\leq$ defined on $\mathbb{R}$ is a full preorder whose corresponding equivalence relation is $\sim$.

(xi) $X \leq Y \quad \Rightarrow \quad X + Z \leq Y + Z$

(xii) $X \leq Y$ and $0 \leq Z \quad \Rightarrow \quad XZ \leq YZ$

Now (x) above tells us that $\sim$ is an equivalence relation. Now we prove it respects all the operations we have defined on L:

L2 If X, Y, X' and Y' are Cauchy sequences and $X \sim X'$ and $Y \sim Y'$ then

(i) $X + Y \sim X' + Y'$

(ii) $-X \sim -X'$

(iii) $XY \sim X'Y'$

(iv) $X \not\sim 0_L$ if and only if $X' \not\sim 0_L$ and then $X^{-1} \sim X'^{-1}$.

(v) $X \leq Y$ if and only if $X' \leq Y'$.

E.3 Construction of The Reals by completion, Step 2

We define $\mathbb{R}$ to be the quotient set: $\mathbb{R} = L/\sim$.

R1 From (L2) it follows that we can define corresponding operations on $\mathbb{R}$ by

Zero	$0_{\mathbb{R}} = [0_L]$
Addition	$[X] + [Y] = [X + Y]$
Negation	$-[X] = [-X]$
One	$1_{\mathbb{R}} = [1_L]$
Multiplication	$[X][Y] = [XY]$
Inversion	If $[X] \neq 0_{\mathbb{R}}$, then $[X]^{-1} = [X^{-1}]$.
Preorder	$[X] \leq [Y] \quad \Leftrightarrow \quad X \leq Y$.

(As usual, the point here is that the results of (L2) mean that the definitions here are independent of the particular members chosen to represent the equivalence classes.)

R2 Applying (R1) to (L1) gives all the basic algebraic laws for $\mathbb{R}$.

(i) $(x + y) + z = x + (y + z)$

(ii) $x + 0 = x$

(iii) $x + (-x) = 0$

(iv) $x + y = y + x$

(v) $(xy)z = x(yz)$

(vi) $x(y + z) = xy + xz$

(vii) $x.1 = x$

(viii) $xy = yx$

(ix) If $x \neq 0$ then $x(x)^{-1} = 1$

(x) The relation $\leq$ defined on $\mathbb{R}$ is a full order.

(xi) $x \leq y \quad \Rightarrow \quad x + z \leq y + z$

(xii) $x \leq y$ and $0 \leq z \quad \Rightarrow \quad xz \leq yz$

R3 We now prove the completeness property of the reals: any Cauchy sequence of real numbers converges to a unique real number. (Note: we are talking here about a sequence of *real* numbers, which is Cauchy in the sense of Equation (1) of this section — ϵ is real also. This is therefore quite a complicated result.)

R4 Finally, $\mathbb{Q}$ is embedded in $\mathbb{R}$ by the map $q \mapsto [q, q, q, \ldots]$. This function is an injection which preserves all the operations and the order. Thus the image of $\mathbb{Q}$ in $\mathbb{R}$ under this mapping is an identical copy of $\mathbb{Q}$ in all salient respects. From now on we may, and often do when it suits us, think of $\mathbb{Q}$ as a this subset of $\mathbb{R}$.

E.4 Construction of The Reals by Dedekind cuts

The basic idea of this method is to obtain a real number a by looking at the set

$$\{\, x \,:\, x \in \mathbb{Q}, x < a \,\}$$

of all rationals less than it; such sets are called *cuts*. We must first define them in a way which does not require prior reference to real numbers.

E.5 Definition: Cut

A *cut* is a subset A of $\mathbb{Q}$ with these properties:

(i) It is initial, that is, if $a \in A, x \in \mathbb{Q}$ and $x \leq a$ then $x \in A$ also.

(ii) Neither A nor its complement $\mathbb{Q} \setminus A$ is empty.

(iii) A does not have a greatest element (that is, it is open).

E.6 Lemma

If A is a cut, then its complement $\mathbb{Q} \setminus A$ has these properties: **(i*)** It is final, that is, if $a \in \mathbb{Q} \setminus A, x \in \mathbb{Q}$ and $x \geq a$ then $x \in \mathbb{Q} \setminus A$ also.

(ii*) Neither $\mathbb{Q} \setminus A$ nor its complement A is empty.

However it may not be open — it may have a least element.

Let us write A^{co} for $\mathbb{Q} \setminus A$ with its least element (if any) removed. Then A^{co} satisfies (i)* and (ii)* above and is open, that is,

(i$^-$) It is final, that is, if $a \in A^{\text{co}}, x \in \mathbb{Q}$ and $x \geq a$ then $x \in A^{\text{co}}$ also.

(ii$^-$) Neither A^{co} nor its complement is empty.

(iii$^-$) A^{co} does not have a least element.

It follows that the set $\{ -b : b \in A^{\text{co}} \}$ is a cut.

Now we can define the reals with their additive structure.

E.7 Definition

The Real Numbers $\mathbb{R}$ is just the set of all cuts in $\mathbb{Q}$, as defined above (there will be no need to form a quotient structure with this approach).

The zero real is $0_{\mathbb{R}} = \mathbb{Q}^-$, the set of all negative rationals.

For any reals (cuts) A and B, $A + B = \{ a + b : a \in A \text{ and } b \in B \}$.

For any real (cut) A, $-A = \{ -b : b \in A^{\text{co}} \}$.

E.8 Lemma

Let A be a cut and r be any rational number > 0. Then there is some $a \in A$ such that $a + r \in A^{\text{co}}$.

Proof. Now construct sequences $a_0, a_1, a_2, \ldots$ and $b_0, b_1, b_2, \ldots$ of rationals inductively as follows:

Choose any $a_0 \in A$ and (since A^{co} is nonempty) $b_0 \in A^{\text{co}}$. Since A is initial and $A^{\text{co}} \subseteq \mathbb{Q} \setminus A$, $a_0 < b_0$. Now, assuming that a_n and b_n have been defined and $a_n \in A$ and $b_n \in A^{\text{co}}$, so that $a_n < b_n$, we define a_{n+1} and b_{n+1}. Consider $\frac{1}{2}(a_n + b_n)$. If $\frac{1}{2}(a_n + b_n) \in A$, define $a_{n+1} = \frac{1}{2}(a_n + b_n)$ and $b_{n+1} = b_n$; if $\frac{1}{2}(a_n + b_n) \in A^{\text{co}}$, define $a_{n+1} = a_n$ and $b_{n+1} = \frac{1}{2}(a_n + b_n)$; otherwise $\frac{1}{2}(a_n + b_n)$ must be the least element of $\mathbb{Q} \setminus A$, in which case define $a_{n+1} = \frac{1}{3}(a_n + b_n)$ and $b_{n+1} = b_n$.

From this definition we can see that every $a_n \in A$, every $b_n \in A^{\text{co}}$ and $0 < b_{n+1} - a_{n+1} \leq \frac{2}{3}(b_n - a_n)$, whence $0 < b_n - a_n \leq (\frac{2}{3})^n (b_0 - a_0)$. Thus we can choose n so that $0 < b_n - a_n \leq r$. Set $a = a_n$, so $a \in A$ and $a + r = a_n + r \geq b_n \in A^{\text{co}}$ so $a_n + r \in A^{\text{co}}$. ■

E.9 Lemma

With these definitions, $\mathbb{R}$ is a commutative group with respect to addition.

Proof. Associativity and commutativity of addition are trivial.

Let A be any cut; we show that $A + 0_{\mathbb{R}} = A$.

First, suppose that $x \in A + 0_{\mathbb{R}}$. Then there is $a \in A$ and $r \in 0_{\mathbb{R}}$ such that $x = a + r$. But then $r < 0$, so $x < a$ and then $x \in A$.

Conversely, suppose that $x \in A$. Since A has no greatest element, there is $b \in A$ such that $a < b$. Then $a = b + (a - b)$ with $b \in A$ and $a - b \in 0_{\mathbb{R}}$, so $a \in A + 0_{\mathbb{R}}$.

Now let us show that $A + (-A) = 0_{\mathbb{R}}$.

First, let $x \in A + (-A)$. Then there are $a \in A$ and $b \in A^{\text{co}}$ such that $x = a - b$. But then $b > a$, so $x < 0$, so $x \in 0_{\mathbb{R}}$.

Conversely, suppose that $x \in 0_{\mathbb{R}}$. Set $x = -r$, where r is a positive rational. Using the lemma above, let $a \in A$ be such that $a + r \in A^{\text{co}}$. Then $-a - r \in -A$ and $x = -r = a + (-a - r) \in A + (-A)$. ■

We now define the order on $\mathbb{R}$. This is easy.

E.10 Definition

For cuts A and B,

$$A \leq B \quad \text{if and only if} \quad A \subseteq B \ .$$

It is easy to check that this is a full order. Also, for any cuts A, B and C,

$$\begin{aligned} \text{if} \quad A \leq B \quad &\text{then} \quad -B \leq -A \\ &\text{and} \quad A + C \leq B + C \ . \end{aligned}$$

E.11 Lemma

For any cuts A, B and C,

(i) If $A \leq B$ then $A + C \leq B + C$.

(ii) If $A < B$ then $A + C < B + C$.

(iii) If $A \leq B$ then $-A \geq B$.

iv If $A < B$ then $-A > B$.

Proof. **(i)** follows trivially from the definition of the order and the others follow easily from this by ordinary group theory. ■

Now we can define the multiplicative structure for non-negative cuts.

E.12 Definition

Suppose that $A \geq 0_{\mathbb{R}}$ and $B \geq 0_{\mathbb{R}}$. Then

$$\begin{aligned} AB &= \{\, ab : a \in A, a \geq 0, b \in B, b \geq 0 \,\} \cup 0_{\mathbb{R}} \ , \\ A^{-1} &= \{\, x : x > 0, x^{-1} \in A^{\mathrm{co}} \,\} \cup \{0\} \cup 0_{\mathbb{R}} \ , \\ 1_{\mathbb{R}} &= \{\, x : x \in \mathbb{Q}, x < 1 \,\} \ . \end{aligned}$$

E.13 Lemma

Let A be a cut, $A > 0_{\mathbb{R}}$, and let r be any rational > 1. Then there is an $a \in A$ such that $ar \in A^{\mathrm{co}}$.

Proof. Since $A > 0_{\mathbb{R}}$, there is $c \in A$ such that $c > 0$. Let $s = c(r-1)$. Then, by the earlier lemma, there is $d \in A$ such that $d + s \in A^{\mathrm{co}}$. Now let $a = \max\{c, d\}$. Since $c, d \in A$, we have $a \in A$. Also $ar = a + a(r-1) \geq a + c(r-1) = a + s \geq d + s$ and $d + s \in A^{\mathrm{co}}$ so $ar \in A^{\mathrm{co}}$. ■

We can now show that the non-negative cuts obey all the usual field laws involving multiplication.

E.14 Lemma

Let A, B and C be cuts, all $\geq 0_{\mathbb{R}}$. Then

(i) $(AB)C = A(BC)$,

(ii) $AB = BA$,

(iii) $A0_{\mathbb{R}} = 0_{\mathbb{R}}$,

(iv) $A(B + C) = AB + AC$,

(v) $A1_{\mathbb{R}} = A$,

(vi) If $A \neq 0_{\mathbb{R}}$ then $AA^{-1} = 1_{\mathbb{R}}$.

Proof. **(i)**, **(ii)** and **(iii)** follow immediately from the definition.

(iv) Given (ii) and (iii) above, this result is trivially true if any of A, B or C are $0_{\mathbb{R}}$; so we assume now that they are all $> 0_{\mathbb{R}}$. Suppose first that $x \in A(B + C)$. If $x < 0$, then $x \in AB + AC$ automatically. Otherwise, there are $a \in A, a \geq 0$ and $y \in B + C, y \geq 0$ such that $x = ay$, and then $b \in B$ and $c \in C$ such that $y = b + c$.

If $b \geq 0$ and $c \geq 0$ we are done, for then $x = ab + ac$ with $ab \in AB$ and $ac \in AC$. Since $b + c = y \geq 0$ we cannot have both b and c negative. We may therefore suppose that $b < 0$ and $c > 0$, the proof in the other case being the same. But then $y < c$, so $y \in C$ and now $x = 0 + ay$ with $0 \in AB$ and $ay \in AC$, so $x \in AB + AC$ as required.

Conversely, suppose that $x \in AB + AC$ and $x \geq 0$. Then $x = ab + a'c$, where $a, a' \in A$, $b \in B$, $c \in C$ and a, a', b and c are all ≥ 0. Let $a'' = \frac{ab+a'c}{b+c}$. This is a weighted average of a and a' (b and c are ≥ 0) and so lies between a and a'; therefore $a'' \in A$ and $a'' \geq 0$. But now $x = a''(b + c) \in A(B + C)$, as required.

(v) Suppose first that $x \in A1_{\mathbb{R}}$ and $x \geq 0$. Then $x = ab$ with $a \in A, a \geq 0$ and $0 \leq b < 1$. But then $0 \leq x \leq a$, so $x \in a$.

Conversely let $a \in A$ with $a \geq 0$. Since A has no greatest element, there is some $b \in A$ with $b > a$. Then $b \neq 0$ and $0 \leq \frac{a}{b} < a$ so $a = b.\frac{a}{b}$ with $b \in A, b \geq 0$ and $\frac{a}{b} \in 1_{\mathbb{R}}, \frac{a}{b} \geq 0$.

(vi) First, let $x \in AA^{-1},\ x > 0$. Then $x = ab$ with $a \in A,\ a \geq 0,\ b^{-1} \in A^{\text{co}}$ and $b > 0$. But then $b^{-1} > a$ so $0 < b < a^{-1}$ and so $ab < aa^{-1} = 1$. Thus $x \in 1_{\mathbb{R}}$.

Conversely, supose that $x \in 1_{\mathbb{R}},\ x > 0$, that is, $0 < x < 1$. Set $r = x^{-1}$ so $r > 1$. By the previous lemma, there is an $a \in A$ such that $ar \in A^{\text{co}}$. Then $(ar)^{-1} \in A^{-1}$ so $a(ar)^{-1} = r^{-1} = x \in AA^{\text{co}}$, as required. ■

E.15 Lemma

For any cuts A, B and C,

(i) If $0_{\mathbb{R}} \leq A \leq B$ and $C \geq 0_{\mathbb{R}}$ then $AC \leq BC$.

(ii) If $0_{\mathbb{R}} < A \leq B$ then $0_{\mathbb{R}} < B^{-1} \leq A^{-1}$.

Proof. **(i)** follows immediately from the definition of the order and **(ii)** follows from that by ordinary ring theory. ■

We can now stop messing around with cuts. The story so far, converting to ordinary notation, is that we have our set $\mathbb{R}$ of reals, with addition, negation and zero defined on $\mathbb{R}$, multiplication and identity defined on the set of non-negative reals and inversion defined on the set of positive reals. These satisfy

For any x, y and z in $\mathbb{R}$,

(Ai) $(x + y) + z = x + (y + z)$

(Aii) $x + y = y + x$

(Aiii) $x + 0 = x$

(Aiv) $x + (-x) = 0$

(Av) If $x \leq y$ then $x + z \leq y + z$ and $-x \geq -y$.

(Avi) If $x < y$ then $x + z < y + z$ and $-x > -y$.

For any non-negative reals x, y and z,

(Mi) $(xy)z = x(yz)$

(Mii) $xy = yx$

(Miii) $x0 = x$

(Miv) $x(y+z) = xy + xz$

(Mv) $x1 = x$

(Mvi) If $x > 0$ then $xx^{-1} = 1$

(Mvii) If $x \le y$ then $xz \le yz$

(Mviii) If $0 < x \le y$ then $0 < y^{-1} \le x^{-1}$.

We now extend the definitions of multiplication and inversion to negative reals.

E.16 Definition

(i) Let x be any real. Then its absolute value $|x|$ is defined in the usual way:

$$|x| \;=\; \begin{cases} x & \text{if } x \ge 0, \\ -x & \text{if } x < 0. \end{cases}$$

(ii) Let x and y be any reals. Then their product xy is defined

$$xy \;=\; \begin{cases} \text{as already defined} & \text{if } x \ge 0 \text{ and } y \ge 0 \\ -x\,|y| & \text{if } x \ge 0 \text{ and } y < 0 \\ -|x|\,y & \text{if } x < 0 \text{ and } y \ge 0 \\ |x|\,|y| & \text{if } x < 0 \text{ and } y < 0 \end{cases}$$

(iii) Let x be any real. Then its inverse x^{-1} is defined:

$$x^{-1} \;=\; \begin{cases} \text{as already defined} & \text{if } x \ge 0, \\ -(-x)^{-1} & \text{if } x < 0. \end{cases}$$

Observe that, because of M(iii) above, the last three displayed equations can be rewritten

$$|x| \;=\; \begin{cases} x & \text{if } x \ge 0, \\ -x & \text{if } x \le 0\ , \end{cases}$$

$$xy \;=\; \begin{cases} \text{as already defined} & \text{if } x \ge 0 \text{ and } y \ge 0 \\ -x\,|y| & \text{if } x \ge 0 \text{ and } y \le 0 \\ -|x|\,y & \text{if } x \le 0 \text{ and } y \ge 0 \\ |x|\,|y| & \text{if } x \le 0 \text{ and } y \le 0 \end{cases}$$

and

$$x^{-1} \;=\; \begin{cases} \text{as already defined} & \text{if } x \ge 0, \\ -(-x)^{-1} & \text{if } x \le 0. \end{cases}$$

It follows from these definitions that $(-x)(-y) = xy$, $x(-y) = (-x)y = -(xy)$ and $(-x)^{-1} = -x^{-1}$ irrespective of the signs of x and y.

We now show that M(i)–M(vii) extend to negative numbers as well.

E.17 Lemma

For any reals x, y and z,

(i) $(xy)z = x(yz)$

(ii) $xy = yx$

(iii) $x0 = x$

(iv) $x(y + z) = xy + xz$

(v) $x1 = x$

(vi) If $x \neq 0$ then $xx^{-1} = 1$

(vii) If $x \leq y$ and $z \geq 0$ then $xz \leq yz$

Proof. First note that, while we must still be careful about multiplicative manipulation, we have already proved that $\mathbb{R}$ is a group with respect to addition and so we can freely use ordinary manipulation of addition, subtraction and zero.

(i), (ii) and **(iii)** all follow immediately from the definitions above.

(iv) We must consider the various cases according as x, y and z are positive or negative. We already know that $x(y + z) = xy + xz$ in the case where x, y and z are all ≥ 0, and we use this freely below.

Suppose now that $x \geq 0, y \geq 0$ and $z \leq 0$. We must consider two subcases, depending on the sign of $y+z$. If $y+z \geq 0$ then, since $-z \geq 0$ also, we have $x(y+z+(-z)) = x(y+z)+x(-z)$ which is $xy = x(y + z) - xz$ and so the required result. In the other subcase, $y + z \leq 0$, we have $-z \geq 0$ and so $xy = x(-z + (y + z)) = x(-z) + x(y + z)$ by the previous subcase $= -xz + x(y + z)$, which gives the required result again.

Suppose now that $x \geq 0, y \leq 0$ and $z \geq 0$; by commutativity of addition, this is the same as the previous case.

Suppose now that $x \geq 0, y \leq 0$ and $z \leq 0$. Then $y + z \leq 0$ so $x(y + z) = -x(-y - z) = -(x(-y) + x(-z)) = -(-xy - xz) = xy + xy$ as required.

This disposes of all cases in which $x \geq 0$. But if $x \leq 0$ we have $x(y + z) = -(-x)(y + z) = -((-x)y + (-x)z)$ by one or other of the cases above, and this $= -(-xy - xz) = xy + xz$ as required.

(v) We already know that $x1 = x$ in the case $x \geq 0$. And if $x \leq 0$ we have $x1 = -((-x)1) = -(-x)$ (since $-x \geq 0$) $= x$ as required.

(vi) We already know that $xx^{-1} = 1$ in the case that $x > 0$. But if $x < 0$ we have $x^{-1} < 0$ also, so $xx^{-1} = (-x)(-x^{-1}) = (-x)((-x)^{-1}) = 1$ as required.

(vii) We already know this in the case $0 \leq x \leq y$. In the case $x \leq 0 \leq y$ we have $xz \leq 0 \leq yz$ and in the case $x \leq y \leq 0$ we have $-y \leq -x$, so $xz = -(-x)z \leq -(-y)z = yz$. ■

This completes the proof of the ordered field structure of $\mathbb{R}$. As usual we observe

E.18 Lemma

The function $\mathbb{Q}$ to $\mathbb{R}$ given by

$$r \;\mapsto\; \{\, x \,:\, x \in \mathbb{Q},\; x < r \,\}$$

is one-to-one and preserves the order and field operations of $\mathbb{Q}$. In other words, this function embeds $\mathbb{Q}$ into $\mathbb{R}$.

And finally, a result which is easy to prove but important enough to be called

E.19 Theorem: Completion

Let X be a nonempty subset of $\mathbb{R}$ which is bounded above. Then it has a least upper bound.

Proof. Let m be the given upper bound. Define $s = \bigcup X$ (the union of this set of reals, thought of as cuts). We will show that s is the required least upper bound.

Firstly we must show that it is in fact a real number, that is, a cut. To see that it is initial, let $r \in s, q \in \mathbb{Q}$ and $q \leq r$. Then there is some member x of X such that $r \in x$. But then x is a cut and so $q \in x$ also; and then $q \in \bigcup X = s$. To see that s is nonempty, note that X is nonempy and every member of X, being a cut, is nonempty. To see that the complement of s is nonempty, note that every member x of X is $\leq m$, that is, $\subseteq m$ and so $s = \bigcup X \subseteq m$ — and the complement of m is nonempty. Finally, s is the union of sets x none of which have a greatest element, so neither does s.

Every member of X is a subset of $\bigcup X$, that is, every member of X is $\leq s$; and that says that s is an upper bound for X.

Suppose that u is any other upper bound for X. That means that every member of X is a subset of u. But then $s = \bigcup X \subseteq u$, that is $s \leq u$. This tells us that s is in fact the least upper bound for X. ■

E.20 Corollary

Every Cauchy sequence in $\mathbb{R}$ converges to a real number.

Proof by intimidation. This is now first year work. ■

F The Complex Numbers, $\mathbb{C}$

The construction of the Complex Numbers is now easy. Since each complex number z can be written uniquely in the form $z = x + \mathrm{i}y$, where x and y are real, it follows that we can construct $\mathbb{C}$ as the set of all pairs $\langle x, y\rangle$ of real numbers with the operations defined appropriately. Since the representation $z = x + \mathrm{i}y$ is unique, there is no need to define an equivalence relation and a quotient set: in this sense the construction of $\mathbb{C}$ is much simpler than the other ones just discussed.

The proper way to define the operations is obvious: the zero and identity are $\langle 0, 0\rangle$ and $\langle 1, 0\rangle$. The sum and product of $\langle a, b\rangle$ and $\langle c, d\rangle$ are $\langle a + c, b + d\rangle$ and $\langle ac - bd, ad + bc\rangle$. The negative and inverse of $\langle a, b\rangle$ are $\langle -a, -b\rangle$ and

$$\langle \frac{a}{\sqrt{a^2 + b^2}}, \frac{-b}{\sqrt{a^2 + b^2}} \rangle.$$

There is no standard order defined on $\mathbb{C}$, so we do not need to worry about constructing one. We can, however, define the absolute value of a complex number $z = x + \mathrm{i}y = \langle x, y\rangle$ to be $\sqrt{x^2 + y^2}$ and then prove that the Complex Numbers are *complete*, that is, that every Cauchy sequence of complex numbers converges (to a complex number).

One of the most of the most important properties of the Complex Numbers is expressed by the *Fundamental Theorem of Algebra*: Every non-constant polynomial (with real or complex coefficients) has at least one (complex) root. A corollary of this theorem is that every polynomial can be factorised completely into linear factors, provided those factors are allowed to have complex coefficients. There are two standard proofs of this theorem; one involves a considerable amount of complex analysis (contour integration and so on), the other a goodly amount of topology (homotopy theory). Both are beyond the scope of these notes (sorry!).

B. ZF SET THEORY

A Zermelo-Fraenkel Axioms for Set Theory

In this appendix I outline the Zermelo-Fraenkel axioms for Set Theory.

The structure of the appendix follows that of Chapter 5 as closely as possible, noting the differences with **VBG** Set Theory as it progresses. In many places the development is identical, and then it is not done all over again here. ▶ 5

The major difference between the two treatments is that in **VBG** Set Theory the basic objects are classes and in **ZF** Set Theory they are sets. There is no direct way of talking about proper classes in **ZF**.

A.1 Definition: Zermelo-Fraenkel Set Theory, ZF

This is a first order theory with equality. It has one binary relation $\in$ and no functions. For simplicity I will use an abbreviation for the "subset" predicate which occurs in these axioms:

$$x \subseteq y \qquad \text{will mean} \qquad (\forall u)(\ u \in x\ \Rightarrow\ u \in y\ )\ .$$

The proper axioms are:—

(ZF1, Extension) $\quad (\forall x)(x \in a \Leftrightarrow x \in b) \quad \Rightarrow \quad a = b$

(ZF2, Specification) Schema: an axiom for every expression $P(x, y_1, y_2, \ldots, y_n)$
$(\forall y_1)(\forall y_2)\ldots(\forall y_n)(\forall a)(\exists w)(\forall x)(x \in w \quad \Leftrightarrow \quad x \in a\ \wedge\ P(x, y_1, y_2, \ldots, y_n))$

(ZF3, Unordered Pairs) $\quad (\forall a)(\forall b)(\exists w)(a \in w\ \vee\ b \in w)$

(ZF4, Unions) $\quad (\forall a)(\exists w)(\forall x)((\exists y)(x \in y\ \wedge\ y \in a) \quad \Rightarrow \quad x \in w)$

(ZF5, Power Set) $\quad (\forall a)(\exists w)(\forall x)(x \subseteq a\ \Rightarrow\ x \in w)$

(ZF6, Infinity) $\quad (\exists w)\big((\exists u)(u \in w\ \wedge\ (\forall x)\neg(x \in u)) \wedge$
$(\forall x)\big(x \in w \quad \Rightarrow \quad (\exists y)(y \in w\ \wedge\ (\forall u)(u \in y \Leftrightarrow u \in x\ \vee\ u = x))\big)\big)$

(ZF7, Formation) Schema: an axiom for every expression $P(x, y)$
$(\forall x)(x \in a\ \Rightarrow\ (\exists! y)P(x, y)) \quad \Rightarrow \quad (\exists w)(\forall y)(y \in w \Leftrightarrow (\exists x)(x \in a\ \wedge\ P(x, y))$

(ZF8, Foundation) $\quad (\forall a)\big(\,(\forall x)(x \in a\ \Rightarrow\ P(x)) \quad \Rightarrow \quad P(a)\,\big) \qquad \Rightarrow \qquad (\forall a)P(a)$

Note the Axiom of Choice is *not* an axiom of **ZF**.

A.2 Sets and classes in ZF

In **ZF** all mathematical objects are sets. As with **VBG**, numbers, functions, relations and so on are all actually constructed as sets (and in much the same way).

Most of the axioms of **ZF** differ from the corresponding ones of **VBG** only in that they make no mention of the predicate SET( ); this is unnecessary since everything in **ZF** is a set anyway.

Even though **ZF** does not allow us to talk about classes in the formal language, it is often highly desirable to use those sorts of ideas and there are various ways of translating notions which we would naturally think of as being about classes into valid **ZF**. The most common way of doing this translation is by replacing the idea of the class by the predicate which defines it. For a simple example, one could write **G** for the "class of all groups" and write $G \in \mathbf{G}$, which can be translated into genuine **ZF** as "G is a group", in the sense "G satisfies the definition of a group". This kind of terminology and notation is rather dangerous and liable to lead one astray; the acid test is whether anything one says using the language of classes can be translated into genuine **ZF**-speak.

Various kinds of class-like construction are acceptable in **ZF**. For example, suppose we write **F** for the class of finite groups and **A** for the class of abelian groups. Then we may write $G \in \mathbf{F} \cap \mathbf{A}$ because we can translate this as G *is a finite abelian group*. For another example, for any group G, its centre, usually written $Z(G)$, is a uniquely defined subgroup and so a group in its own right. Thus we can describe Z as a "function $\mathbf{G} \to \mathbf{G}$". This is perfectly proper, provided we take this to be definition of a function by description, as in Definition **4**.A.5 — the definition of a function as a class, as in **5**.B.16 is not available to us. This can be quite a nuisance in areas of mathematics which make a lot of use of ideas which are most naturally expressed as functions between classes: functors in algebraic topology, universal algebra and more generally category theory spring to mind.

► **4**.A.5
► **5**.B.16

The kind of argument which must not be used in **ZF** is one which quantifies classes, for instance one which talks about all classes with some given property. The reason this is outlawed is that, in trying to translate this using the predicates which define the classes, one ends up talking about all predicates of such and such a form, and this is just not part of first-order logic. For example, there is no way at all of expressing the idea of "for all functions $\mathbf{G} \to \mathbf{G}$" in **ZF**. It is always possible, of course, that one might be able to replace the argument with another completely different one which is valid **ZF** and does come to the same conclusion, however this is not automatically possible.

The Axiom of Foundation provides an example of straightforward translation. As expressed in **VBG**, the axiom mentions a class w which turns out to be a proper class; indeed $w = \mathbf{U}$. This cannot be said in this way in (formal) **ZF**, so the **ZF** version of the axiom uses the usual way of getting around this by using a predicate instead. The idea of the "class of all sets" is not directly expressible in **ZF**, however a construction of the form "for all sets a, such and such is true" is.

By far the most important difference in the axioms is in the Axiom of Specification. In **VBG** the axiom tells us that, for any predicate $P(x)$ there is a corresponding class $\{\, x : P(x)\}$. In **ZF** the corresponding axiom is much more limited; it tells us that, *for any set* a and any predicate $P(x)$ there is a corresponding *set* $\{\, x : x \in a \ \wedge \ P(x)\}$.

B The first six axioms

B.1 ZF1: The Axiom of Extension

As with **VBG**, the opposite implication is given to us by Substitution of Equals, and so this axiom tells us that

$$A = B \quad \text{if and only if} \quad (\forall x)(x \in A \Leftrightarrow x \in B) \ ,$$

in other words, two sets are equal if and only if they have the same members.

B.2 ZF2: The Axiom of Specification and the Empty Set

Is there any guarantee that there are any sets at all? The Axiom of Specification does not help with this basic question, since is now requires a set before stating the existence of another one. However an appeal to the Axiom of Infinity guarantees the existence of a set, just as with **VBG**.

To define the empty set, we may proceed as follows. Define it as the set $\varnothing$ with this property:

$$x \in \varnothing \ \Leftrightarrow \ x \neq x \ .$$

From this it is easy to deduce the usual properties of the empty set, such as the alternative definition

$$(\forall x) \neg (x \in \varnothing)$$

all except for the fact that it exists at all. To show that the empty set exists, appeal to the mysterious set, w say, which the Axiom of Infinity tells us exists; then the Axiom of Specification tells us that the set $\{ x : x \in w \ \wedge \ x \neq x \}$ exists also. Then show that this set is in fact the empty set.

The notation $\{ x : P(x) \}$ can be used fairly freely in **ZF**, however it is essential that the expression $P(x)$ implies that x is a member of some already-known set, otherwise the construction is meaningless in **ZF**. For example, the construction

$$L \ = \ \{ x : x \in \mathbb{R} \ \wedge \ x < 0 \}$$

is OK, whereas

$$\mathcal{G} \ = \ \{ G : G \text{ is a group} \}$$

is not.

B.3 Some useful notation (definitions)

The definitions of $x \notin A$, $A \subseteq B$, $A \subset B$, $A \supseteq B$, $A \supset B$, $(\forall x \in A) P(x)$ and $(\exists x \in A) P(x)$ are exactly the same as for **VBG**, however of course now everything must be a set.

B.4 Properties of subsets

The following are now easily proved

(i) The empty set is a subset of every set: $\varnothing \subseteq A$, and the only subset of the empty set is itself: $A \subseteq \varnothing \quad \Leftrightarrow \quad A = \varnothing$.

(ii) The subset relation has the properties of a partial order.

B.5 The Universe

There is no Universe. Sorry about that.

(The Russell Paradox tells us that there is no set of all sets.)

B.6 ZF5: The Axiom of Power Sets

This axiom tells us that, for any A, the power set $\mathcal{P}(A) = \{ X : X \subseteq A \}$ exists.

Because this is **ZF**, A must be a set and then so is $\mathcal{P}(A)$. As with **VBG**, the axiom, as stated, only says that, for any set A, there is a set, W say, which contains all the subsets of A as members (that is, $X \subseteq A \ \Rightarrow \ X \in W$). To see that the power set P itself exists, the argument is slightly simpler than the one needed in **VBG**: the Axiom of Specification can be used directly to define

$$P \ = \ \{ X \ : \ X \in W \ \wedge \ X \subseteq A \}$$

B.7 ZF3: The Axiom of Unordered Pairs

This axiom tells us that, for any a and b, the unordered pair $\{a, b\} = \{ x : x = a \ \vee \ x = b \}$, exists. It follows that the singleton $\{a\} = \{ x : x = a \}$ exists also, since $\{a\} = \{a, a\}$.

As usual, a and b must be sets and then so are $\{a, b\}$ and $\{a\}$. And once again, the usual properties follow easily.

B.8 ZF4: The Axiom of Unions

The union of two sets and the union of a set of sets are defined in the same way as for **VBG**.

$$A \cup B \ = \ \{ x : x \in A \ \text{ or } \ x \in B \} \qquad \text{and} \qquad \bigcup \mathcal{A} \ = \ \{ x : (\exists W)(x \in W \ \text{ and } \ W \in \mathcal{A} \}$$

Axiom ZF4 tells us that, for any $\mathcal{A}$, $\bigcup \mathcal{A}$ exists. As usual, since this is **ZF**, $\mathcal{A}$ must be a set and then so is $\bigcup \mathcal{A}$. (Note that, in **VBG**, the existence of the union of classes is given by the Axiom of Specification. However in **ZF**, we need Axiom ZF4 in order to be able to talk about unions at all.) It follows that the union of two sets, and hence the union of any finite number of sets, is also defined.

B.9 Intersections

The definition of intersections of sets is rather more tricky than is the case with classes in **VBG**. the intersection of two sets can be defined by

$$A \cap B \ = \ \{ x \ : \ x \in A \ \wedge \ x \in B \}.$$

The intersection of any (nonzero) finite number of sets can be defined similarly, and then the usual properties are easily proved.

The intersection of any nonempty set of sets can be defined in the usual way: if $\mathcal{A}$ is a nonempty set of sets, then its intersection is:

$$\bigcap \mathcal{A} \ = \ \{ x : (\forall W \in \mathcal{A})(x \in W) \}$$

Here is where we must be careful: it is not enough to simply write this definition down. The axioms of **ZF** do not guarantee the intersections existence as easily as that, because the definition does not conform to the **ZF** form of the Axiom of Specification. So we must verify that the intersection exists and is unique.

For a start, if $\mathcal{A}$ is empty, this intersection is in fact undefined. If it existed, then the definition can be seen to be equivalent to that of the universe, which does not exist in **ZF**.

If $\mathcal{A}$ is nonempty, then it must contain a member, A say. Then we observe (with a small proof) that the definition above is equivalent to

$$\bigcap \mathcal{A} \;=\; \{\, x : x \in A \;\wedge\; (\forall W \in \mathcal{A})(x \in W) \,\}$$

which does conform to the **ZF**Axiom of Specification. Then finish off by checking uniqueness.

B.10 The calculus of sets

We have to be more careful with the calculus of sets than is the case with **VBG**.

For a start, the complement of a set is undefined.

For two sets A and B, the *set difference* $A \setminus B$ is defined in the same way as in **VBG**: $A \setminus B = \{\, x : x \in A \;\wedge\; x \notin B \,\}$, and the definition is also good in **ZF**. The usual properties are easily proved.

The union and intersection of two sets are defined as discussed above and have all the usual properties.

B.11 Ordered pairs

The primitive ordered pair $\langle a, b \rangle$ of two sets is defined in exactly the same way as for **ZF**. Its properties, and the consequent definition of the (ordinary) ordered pair go exactly as for **VBG**.

Since we cannot speak of classes in **ZF**, there is no need for the more general definition of an ordered pair which is used in **VBG** to define a pair of classes.

B.12 Cartesian products

The cartesian product is defined, and its properties proved, in the same way as for **VBG**.

Note that the horrid little proof given in Chapter **5** that the cartesian product of two sets is a set works just as well in **ZF**, but here is used to tell us that the cartesian product exists.

► Ch.5

B.13 Relations, functions, cartesian powers and sequences

Everything here is exactly the same as for **VBG**, except of course that everything in sight must be a set.

C The Natural Numbers

This chapter holds good in **ZF** with hardly any changes.

▶ **5**.D.2 In Definition **5**.D.2 the bald definition

$$\mathbb{N} = \{\, n : n \text{ is a member of every inductive set} \,\}.$$

won't do. Instead we show that there exists a unique set $\mathbb{N}$ with the property that

$$\text{for all } x, \quad x \in \mathbb{N} \text{ if and only if } n \text{ is a member of every inductive set.}$$

Noting that the set W of the axiom is not unique, we show that $\mathbb{N}$ exists by using the fact that the axiom states that at least one such set W exists, and so there exists a set

$$\mathbb{N} = \{\, n : n \in W \text{ and } n \text{ is a member of every inductive set} \,\}.$$

We then show (easy) that $\mathbb{N}$ has the property stated above. Then it is also easy to prove that it is unique.

▶ **5**.D.4 In Definition **5**.D.4, of course, we only define a transitive *set*. This is not a problem for the rest of the section.

D Well ordering

and ...

E The Axiom of Choice

and ...

F The Axioms of Foundation and Formation

▶ Ch.**5** No surprises here. Everything in these sections of Chapter **5** holds good in **ZF** word for word — so long as references to classes are replaced by references to sets wherever they appear.

▶ Ch.**6** The same remarks apply to the treatment of Cardinality in Chapter **6**.

Dealing with ordinal numbers when one cannot talk about classes at all can be done, but can be a bit tiresome. In the next section I will present the main results of the section on
▶ Ch.**6** Ordinal numbers in Chapter **6** translated into **ZF**-speak.

G Ordinal numbers

G.1 Definition

An ordinal number is a set α such that

(i) α is transitive (that is, every member of α is also a subset of α).

(ii) For all $x, y \in \alpha$, one of the following hold: $x \in y$, $x = y$ or $y \in x$.

Note that (i) above is equivalent to

(i′) $x \in \alpha \;\Rightarrow\; x \subset \alpha$.

G.2 Ordering an ordinal number

Let α be an ordinal. We define the relation $\leq$ on α by

$$x \leq y \qquad \text{if and only if} \qquad x \in y \quad \text{or} \quad x = y \ .$$

This relation is in fact a well order on α.

G.3 Lemma

If two ordinal numbers are order-isomorphic, then they are equal.

G.4 Lemma

Every initial segment of an ordinal number is an ordinal number.

Therefore every member of an ordinal number is an ordinal number.

G.5 Theorem

(i) If α is an ordinal number and $x \in \alpha$ then x is an ordinal number also.

(ii) If α and β are ordinal numbers then one of $\alpha \in \beta$, $\alpha = \beta$ or $\beta \in \alpha$ must hold.

(iii) Every nonempty set of ordinal numbers has a least element.

G.6 Remark

This theorem tells us that, if there were such a thing as the set of all ordinal numbers, then it would itself be an ordinal number, and thus a member of itself. Therefore ...

G.7 Theorem

There is no such thing as the set of all ordinal numbers.

G.8 Theorem

Every well ordered set is order-isomorphic to exactly one ordinal number.

G.9 Definition

If A is a well ordered set, then the unique ordinal number to which it is order-isomorphic is called the *order type* of A.

G.10 Example

$\mathbb{N}$ is an ordinal number.

G.11 Remark

In the context of ordinal numbers, the set $\mathbb{N}$ is usually denoted ω.

G.12 Example

Every natural number is an ordinal number.

G.13 Theorem

If α is an ordinal, then so is α^+.

G.14 Examples

As examples of the last theorem, the following are ordinal numbers.

$$\omega^+ = \{0, 1, 2, \ldots\} \cup \{\omega\}$$

$$\omega^{++} = \{0, 1, 2, \ldots\} \cup \{\omega, \omega^+\}$$

$$\omega^{+++} = \{0, 1, 2, \ldots\} \cup \{\omega, \omega^+, \omega^{++}\}$$

and so on.

G.15 Theorem

Let A be a set of ordinals. Then $\bigcup A$ is an ordinal.

C. SOME ALGORITHMS

In this section I explain in more detail the algorithms required for some of the constructions in Chapter **9**.

▶ Ch.**9**

A Preliminaries

A.1 Symbols and their arities

Here are some basic functions, defined more for convenience than anything else. As presented here, I am assuming that the functions and relations are as given in Section **4**.B.1 and no others, with the following Gödel numbers assigned; for a system **S** with more functions and relations, make the obvious changes.

▶ **4**.B.1

$\neg$	0	$\bar{0}$	6	
$\Rightarrow$	1	s	7	(the successor function)
$\forall$	2	$+$	8	
(	3	$\times$	9	
)	4	$=$	10	
,	5	v_i	$11 + i$	(i^{th} variable symbol)

First some functions which test for various kinds of symbols from their Gödel numbers.

```
isNotSym(n) { if (n=0) return 1; else return 0 ; }

isImpSym(n) { if (n=1) return 1; else return 0 ; }

isForAllSym(n) { if (n=2) return 1; else return 0 ; }

isOpenSym(n) { if (n=3) return 1; else return 0 ; }

isCloseSym(n) { if (n=4) return 1; else return 0 ; }

isComma(n) { if (n=5) return 1; else return 0 ; }

isFunSym(n) { if (3≤n≤9) return 1; else return 0 ; }

isRelSym(n) { if (n=10) return 1; else return 0 ; }

isVarSym(n) { if (n≥11) return 1; else return 0 ; }
```

and a function to return arities (for both functions and relations)

```
1  arity(n) {
2     if (n=6) return 0;
3     if (n=7) return 1;
4     if (8 ≤ n ≤ 10) return 2;
5     return 0;
6  }
```

Note that if `n` is not the Gödel number of any function or relation symbol this function returns 0; this is never used, but we do this so that the function is recursive, not partial.)

► **9**.A.2 In what follows we will also use the functions defined in Lemma **9**.A.2.

Some of the functions we are about to look at can best be thought of as "scanning" a string, checking out symbols and substrings as they go. Typically they will work with two numbers, `x`, the Gödel number of the string and `p`, the index of the symbol currently being looked at in the string. Some functions will check out a substring (usually a subexpression) in some way; typically such a function will return the index of the next symbol after the substring, making it easy for the function that called it to proceed.

A.2 String manipulations

We will need some functions which perform basic manipulations with strings.

Firstly, to get the substring of a string; given a string with Gödel number `x`, the function `Substring(p,q,x)` finds the substring which extends from its $\mathtt{p}^{\text{th}}$ to its $\mathtt{q}^{\text{th}}$ entry (inclusive) and returns the Gödel number of that substring. It assumes that $1 \leq \mathtt{p} \leq \mathtt{q} \leq \mathtt{len(x)}$ (we will only be using it when these inequalities hold, so we are not interested in what the function does in other cases).

```
1  Substring(p,q,x; z,r) {
2     z := ent(q,x);
3     r := q-1;
4     while (r ≥ p) {
5        add(ent(r,x),z);
6        r := r-1;
7     }
8  return z;
9  }
```

The substring will be built up in z, one symbol at a time, starting from the right hand end of the substring in x.

We will need to concatenate two strings. Given strings with Gödel numbers `x` and `y`, the function `Concat(x,y)` returns the Gödel number of the concatenated string `x` followed by `y`.

```
1  Concat(x,y; z,p) {
2     z := y;
```

The new string will be built up in z,

```
3     p:=len(x);
4     while (p ≥ 1) {
5        add(ent(p,x),z);
6        p := p-1;
7     }
8  return z;
9  }
```

starting with y, then adding one symbol at a time from x.

We will also need to replace a single entry by a substring. Given a string with Gödel number `x`, an index `p` and another string `s`, the function `Replace(p,x,s)` replaces the $\mathtt{p}^{\text{th}}$ entry of `x` with the string `s` and returns the Gödel number of the resulting string. We do this by using the `Substring` function to pull `x` apart and then the `Concat` function to reassemble it differently.

```
1  Replace(x,p,s; ) {
2     if (len(x)=1) return s;
3     if (p=1) return Concat(s,Substring(2,len(x),x));
4     if (p=len(x)) return Concat(Substring(1,len(x)-1,x),s);
5     else return Concat(Concat(Substring(1,p-1,x),s),Substring(p+1,len(x),x));
6  }
```

B Algorithms for Gödel's Theorem

B.1 Recognising terms

Here we set about defining a function which will decide whether a given string is a valid expression or not — more precisely, it decides whether a given number `x` is the string Gödel number of a valid expression or not.

Before doing that we need a function isTerm which will decide whether a given substring is a term or not.

More correctly: the function `isTerm(x,p)` will test to see if the $\mathtt{p}^{\text{th}}$ entry of the string with Gödel number `x` is the first character of a valid term or not. It returns 1 for yes and 0 for no.

We also want a function which simply skips a term. Given the Gödel number `x` of an expression and the index `p` of an entry in it, `SkipTerm(x,p)` will return the index of the first entry *after* that term. We will only use this function when it is already known that `(x,p)` is indeed the start of a valid term.

For example, consider the distributive law $(\forall z)(\forall y)(\forall z)(\,(x+y)z = xz + yz\,)$ written out in its fully formal form:

$$(\forall x(\forall y(\forall z = (\times(+(x,y),z),+(\times(x,z),\times(y,z))))))$$

and let us suppose that its Gödel number is `x`. In this there is a term $\times(+(x,y),z)$ whose first character (the $\times$) is the 12th entry in the expression, so our functions should give

```
isTerm(x,12) = 1
SkipTerm(x,12) = 23
```

since the first entry after this term is the 23rd (a comma). Similarly, its 8th entry (a $\forall$ symbol) is not the first character of a term, so

```
isTerm(x,8) = 0
SkipTerm(x,8) = Don't care.
```

Here are the functions:

```
1  SkipTerm(x,p; ch,ar,count) {
2     ch := ent(x,p);

3     if (isVarSym(ch) return p+1;

4     if (isFunSym(ch)) {
5        ar := arity(ch) ;
6        p := p+2
7        count := 0;
8        while (count < ar) {
9           p := SkipTerm(x,p);
10          count := count + 1 ;
11          p:=p+1 ;
12       }
13       return p;
14    }

15    return 0;
16 }
```

and

```
1  isTerm(x,p; ch,ar,count) {
2     ch := ent(x,p);                          Get the first character.
3     if (isVarSym(ch) return 1;               If it is a variable, OK.
4     if (isFunSym(ch)) {                      If it is a function symbol ...
5        ar := arity(ch) ;                     Get its arity.
6        p := p+1 ; ch = ent(x,p);             Get the next character.
7        if (not isOpenSym(ch)) return 0;      Should be an opening parenthesis.
8        p := p + 1; ch := ent(x,p);           Get the next character.
9        count := 0;                           Start counting arguments.
10       while (count < ar) {                  Loop for correct number of args.
11          if (not isTerm(x,p)) return 0;     Arg should be a term.
12          p := SkipTerm(x,p);                If not return 0.
13          count := count + 1 ;               If OK, update the count.
14          ch := ent(x,p) ;                   Get then next character.
15          if (count < ar) {                  For all but the last entry,
16             if (not isComma(ch)) return 0;        this should be a comma.
17             p := p + 1; ch := ent(xp);
18          }
19       }
20       if (isCloseSym(ch)) return 1;          Now we've looked at all the args.
21       else return 0 ;                        Next character should be a ).
22    }
23    return 0;                                 If we get here, the first character was not
24 }                                             a variable or a function; not OK.
```

Shortly we will need a function which gets a term: given the Gödel number `x` of an expression and the index `p` of the first entry of a term in it, it returns the Gödel number of that term as a string.

```
1  GetTerm(x,p; q) {
2     q := SkipTerm(x,p)) - 1;
3     return Substring(p.q.x);
4  }
```

B.2 Recognising expressions (for 9.A.4)

▶ 9.A.4

Next we define a very similar pair of algorithms for checking subexpressions. The function `isSubxpression(x,p)` will test to see if the p^{th} entry of the string with Gödel number `x` is the first character of a valid subexpression or not, returning 1 for yes and 0 for no. The function `SkipSubexpression(x,p)` will simply skip the subexpression starting at `(x,p)`.

Recalling that an expression must be of one of the forms

- $r(t_1, t_2, \ldots, t_n)$ where r is an n-ary relation symbol and $t_1, t_2, \ldots, t_n$ are terms,
- $(\neg P)$ where P is an expression,
- $(P \Rightarrow Q)$ where P and Q are expressions,
- $(\forall x P)$ where x is a variable symbol and P is an expression,

we have:

```
1  SkipSubexpression(x,p; ch,ar,count) {
2     ch := ent(x,mu);

3     if (isRelSym(ch)) {
4        ar := arity(ch) ;
5        p := p+2;
6        count := 0;
7        while (count < ar) {
8           p := SkipTerm(x,p))+1;
9           count := count + 1;
10          }
11       }
12       return p;
13    }

14    p:=p+1; ch:=ent(x,p);

15    if (isNotSym(ch)) {
16       p := p+1;
17       return := SkipSubxpression(x, p))+1;
18    }

19    if (isForAllSym(ch)) {
20       p := p+2;
21       p := SkipSubexpression(x,p));
22       return p+1;
23    }

24    p := p+1;
25    p := SkipSubexpression(x,p));
26    p := p+1;
27    p := SkipSubexpression(x,p);
28    return p+1;
29 }
```

and

```
1  isSubexpression(x,p; ch,ar,count) {
2     ch := ent(x,mu);                          Get the first character.

3     if (isRelSym(ch)) {                       Starts with a relation symbol.
4        ar := arity(ch) ;                      This section is almost exactly the same
5        p := p+1; a := ent(x,p);                  as the isTerm function above.
6        if (not isOpenSym(ch)) return 0;
7        p := p+1; ch := ent(x,p);
8        count := 0;
9        while (count < ar) {
10          if (not isTerm(x,p)) return 0;
11          p := SkipTerm(x,p);
12          count := count + 1;
13          ch := ent(x,p) ;
14          if (count < ar) {
15             if (not isComma(ch)) return 0;
16             p := p + 1; ch := ent(x,p);
17          }
18       }
19       if (not isCloseSym(ch)) return 0;
20       return 1 ;
21    }

22    if (not isOpenSym(a)) return 0;           All other forms must start with an
23    p := p + 1; ch := ent(x,p);                  opening parenthesis.

24    if (isNotSym(ch)) {                       A Not symbol.
25       p := p+1; ch := ent(x,p);              Get next character.
26       if (not isSubexpression(x,p)) return 0; It should start a subexpression.
27       p := SkipSubexpression(x,p);
28       ch := ent(x,p);                        Get next character.
29       if (not isCloseSym(ch)) return 0;      It should be a closing parenthesis.
30       return 1;                              If we got to here, all is OK.
31    }

32    if (isForAllSym(ch)) {                    A For All symbol.
33       p := p + 1; ch := ent(x,p);            Get next character.
34       if (not isVarSym(ch)) return 0;        It should be a variable symbol.
35       p := p + 1; ch := ent(x,p);            Get the next character.
36       if (not isSubexpression(x,p)) return 0;    It should start a subexpression.
37       p := SkipSubexpression(x,p);
38       ch := ent(x,p);                        Get the next character.
39       if (not isCloseSym(ch)) return 0;      It should be a closing parenthesis.
40       return 1;                              If we got to here, all is OK.
41    }

42    if (not isSubexpression(x,p)) return 0;   Current character should start a subexpression.
```

```
43    p := SkipSubexpression(x,p);
44    ch := ent(x,p);                          Get next character.
45    if (not isImpSym(ch)) return 0;          It should be an implication symbol.
46    p := p + 1; ch := ent(x,p);              Get next character,
47    if (not isSubexpression(x,p)) return 0;      It should start a subexpression.
48    p := SkipSubexpression(x,p);
49    ch := ent(x,p);                          Get next character.
50    if (not isCloseSym(ch)) return 0;        It should be a closing parenthesis.
51    return 1 ;                               If we got to here, all is OK.
52 }
```

Now it is easy to make a function `isExpression(x)`, which tests whether an entire string is a valid expression or not; more correctly, it tests whether its argument is the string Gödel number of an expression or not.

```
1  isExpression(x; p) {
2     p = isSubexpression(x,1));
3     if (p = len(x)+1) return 1;
4     else return 0;
5  }
```

We will need a function which gets a subexpression: given the Gödel number `x` of an expression and the index `p` of the first entry of a subexpression, it returns the Gödel number of that subexpression.

```
1  GetSubexpression(x,p; q) {
2     q := SkipSubexpression(x,p)) - 1;
3     return Substring(p.q.x);
4  }
```

B.3 Recognising sentences (for 9.A.5)

▶ 9.A.5

Next we wish to write a function to test if a given string is a sentence. To do this, we first write a function which decides whether a given variable occurs free in a given expression. And before that in turn we write a function to decide whether a given variable occurs in a term or not.

$$\texttt{varInTerm(x,p,v)} = \begin{cases} 1 & \text{if the term that starts at character number } \texttt{p} \text{ in the string with} \\ & \text{Gödel number } \texttt{x} \text{ contains the variable with Gödel number } \texttt{v}, \\ 0 & \text{otherwise.} \end{cases}$$

We will only use this function when we already know that `x` is the Gödel number of a valid expression, that `p` is the index of the first character of a valid term in that expression and that `v` is the Gödel number of a variable symbol; we do not care what nonsense the function may get up to in any other case.

```
1  varInTerm(x,p,v; ch,arity,count) {
2     ch := ent(x,p);                          Get first character.
3     if (ch=v) return 1;                      If it's a variable, we are done.
4     ar := arity(ch);                         Must be function symbol; get arity.
5     p := p+2; count := 0 ;                   Skip the parenthesis; get next character.
6     while (count < ar) {                     Count through the arguments.
7        if (varInTerm(x,p,v)=1) return 1;     Does v occur in the argument?
8        p := SkipTerm(x,p);
9        count:=count+1; p:=p+1;               Skip the comma; get next character.
10    }
11    return 0;                                If we get to here something's wrong.
12 }
```

Now we can test whether a variable occurs free in an expression. Referring to the definition of binding in **3**.A.9 we see that the variable v occurs free in an expression if one of the following cases obtains:

▶ 3.A.9

- The expression is atomic, $r(t_1, t_2, \ldots, t_n)$ say, and v occurs in one of the terms $t_1, t_2, \ldots, t_n$.
- The expression is of the form $(\neg P)$ and v occurs free in P.
- The expression is of the form $(P \Rightarrow Q)$ and v occurs free in at least one of P or Q.
- The expression is of the form $(\forall x P)$, v is not x and v occurs free in P.

We define a function `FreeInSubexpression(x,p,v)`; if `x` is the string Gödel number of a valid expression, `p` is the index of a character in that expression which starts a valid subexpression and `v` is the Gödel number of a variable symbol, the function will return 1 for yes if that variable occurs free in that subexpression and 0 for no otherwise.

```
FreeInSubexpression(x,p,v; ch,ar,count) {
   ch := ent(x,p);

   if (isRelSym(ch)) {
      p := p+2; count := 0 ; ar = arity(ch);
      while (count < ar) {
         if (varInTerm(x,p,v)=1) return 1;
         SkipTerm(x,p);
         count := count+1; p := p+1;
      }
      return 0;
   }

   p := p+1; ch := ent(x,p) ;

   if (isNotSym(ch)) {
      p := p+1;
      return FreeInSubexpression(x,p,v);
   }

   if (isForAllSym(ch)) {
      p:=p+1; ch:=ent(x,p) ;
      if (ch=v) return 0;
      p:=p+1;
      return FreeInSubexpression(x,p,v);
   }

   p := p+1;
   if (freeInSubexpression(x,p,v)=1) return 1;
   p:= p+1;
   if (freeInSubexpression(x,p,v)=1) return 1;
   else return 0;

}
```

It is now easy to write a function which checks whether a variable occurs free in a whole expression or not. If `x` is the string Gödel number of a valid expression and `v` is the Gödel number of a variable symbol, `freeInExpression(x,v)` returns 1 for yes if that variable occurs free in the expression and 0 for no otherwise.

```
freeInExpression(x,v) {
   return freeInSubexpression(x,1,v);
}
```

And now we can write a function which checks if a string `x` is a valid sentence or not. This is fairly easy. First check that it is an expression, then check that it has no free variables by checking that variable symbols with Gödel numbers up to `x` do not occur free in it . If nothing has gone wrong up to there, it is a sentence, so return 1.

(Referring back to the original definition of the function P, we see that, for all x and y, $P(x,y) \geq x$ and $P(x,y) \geq y$ and that P is strictly increasing in both variables. It follows that the Gödel number of a string is greater than the Gödel number of any of its substrings and greater than the Gödel number of any of its characters. So, in order to test whether the expression of Gödel number `x` contains any free variables or not it is enough to test it for free occurrences of all variables of Gödel number from 11 up to `x`. Why 11? Because that is the Gödel number of the first variable symbol. This is obviously not very efficient, but we are not interested in efficiency here, and it is simple.)

```
1  isSentence(x; p,w,ch,v) {
2     if (not isExpression(x)) return 0;
3     v:=11;
4     while(v ≤ x) {
5        if (isVarSym(v) & FreeInExpression(x,v)) return 0;
6        v := v+1;
7     }
8     return 1;
9  }
```

B.4 Substitution

Given the Gödel numbers `x` of an expression, `v` of a variable symbol and `t` of a term, the function `SubstInTerm(x,v,t)` returns the Gödel number of the expression made by the substitution `x[v/t]`.

```
1  SubstInTerm(x,v,t; ch,p,ar,y,s) {
2     ch := ent(1,x);
3     if (ch=v) return t;
4     if (len(x) = 1) return x;
5     ar := arity(ch);
6     y := Substring(x,1,2);
7     p := 3; count := 1;
8     while (count ≤ ar) {
9        s := GetTerm(x,p);
10       y := Concat(y, SubstInTerm(s,v,t));
11       p := SkipTerm(x,p);
12       y := Concat(y, Substring(x,p,p));
13       p := p+1; count := count + 1;
```

```
14    }
15    return y;
16 }
```

Now we can write a function which will make a similar substitution in an expression. The code is lengthy because it has to deal with the various ways an expression can be constructed, but the logic is very similar to that of the last function.

```
1  SubstInExpression(x,v,t; ch,p,ar,y,s) {
2     ch := ent(1,x);

3     if (isRelSym(ch)) {
4        ar := arity(ch);
5        y := Substring(x,1,2);
6        p := 3; count := 1;
7        while (count ≤ ar) {
8           s := GetTerm(x,p);
9           y := Concat(y, SubstInTerm(s,v,t));
10          p := SkipTerm(x,p);
11          y := Concat(y, Substring(x,p,p));
12          p := p+1; count := count + 1;
13       }
14       return y;
15    }

16    ch := ent(2,p);

17    if (isNotSym(ch)) {
18       y := Substring(x,1,2);
19       p := 3;
20       s := GetSubexpression(x,p);
21       y := Concat(y, SubstInExpression(s,v,t));
22       p := SkipSubexpression(x,p);
23       y := Concat(y, Substring(x,p,p);
24       return y;
25    }

26    if (isForAllSym(ch)) {
27       ch := ent(x,3);
28       if (ch = v) return x;
29       y := Substring(x,1,3);
30       p := 4;
31       s := GetSubexpression(x,p);
32       y := Concat(y, SubstInExpression(s,v,t));
33       p := SkipSubexpression(x,p);
34       y := Concat(y, Substring(x,p,p));
```

```
35          return y;
36      }

37      y := Substring(x,1,1);
38      p := 2;
39      s := GetSubexpression(x,p);
40      y := Concat(y, SubstInExpression(s,v,t));
41      p := SkipSubexpression(x,p);
42      y := Concat(y, Substring(x,p,p));
43      p := p+1;
44      s := GetSubexpression(x,p);
45      y := Concat(y, SubstInExpression(s,v,t));
46      p := SkipSubexpression(x,p);
47      y := Concat(y, Substring(x,p,p));
48      return y;

49  }
```

B.5 Acceptability

Testing for acceptability is not difficult. We use the inductive definition: the substitution $[v/t]$ is acceptable in ...

- any atomic expression $r(s_1, s_2, \ldots, s_n)$ always;
- $(\neg P)$ iff it is acceptable in P;
- $(P \Rightarrow Q)$ iff it is acceptable in both P and Q; and
- $(\forall u P)$ iff it is acceptable in P and, if P contains v free, t does not contain u.

```
1  Acceptable(x,v,t; ch,p,ar,y,s) {
2     ch := ent(1,x);

3     if (isRelSym(ch)) return 1;
4     }

5     ch := ent(2,p);

6     if (isNotSym(ch)) {
7        y := GetSubexpression(x,3);
8        return Acceptable(y,v,t);
9     }

10    if (isForAllSym(ch)) {
11       u := ent(3,x);
12       y := GetSubexpression(x,5);
13       if (not Acceptable(y,v,t)) return 0;
14       if (FreeInSubexpression(x,5,v) and
15                       (VarInTerm(t,1,v)) return 0;
16       else return 1;
17    }

18    y := GetSubexpression(x,2);
19    if (not Acceptable(y,v,t)) return 0;
20    p := SkipSubexpression(x,p)+1;
21    y := GetSubexpression(x,2);
22    if (not Acceptable(y,v,t)) return 0;
23    else return 1;

24 }
```

B.6 Recognising axioms (for 9.A.6)

▶ 9.A.6

First we test a string to see if it is an instance of Axiom PL1. The function isPL1(**x**) returns 1 if string **x** is an instance of Axiom PL1, 0 otherwise. This axiom in its formal form is $(P \Rightarrow (Q \Rightarrow P))$, where P and Q are expressions.

In this implementation we first check that **x** is the Gödel number of a valid expression. Then the main part of the listing is concerned with scanning the expression in the now-familiar way to check that it is of the form $(P \Rightarrow (Q \Rightarrow R))$, remembering where the subexpressions P and Q start (π and ρ) and the length λ of P. Finally (Line 21) it checks that P and R are equal.

```
1  isPL1(x; p,ch,P1,P2) {
2     if (not isExpression(x) return 0;
3     p := 1; ch := ent(x,p);
4     if (not isOpener(ch)) return 0;
5     p := p+1;
6     if (not isSubexpression(x,p)) return 0;
7     P1 := GetSubexpression(x,p);
8     p := SkipSubexpression(x,p);
9     p := p+2;
10    p := skipSubexpression(x,p);
11    p := p+1;
12    P2 := GetSubexpression(x,p);
13    if (P1 ≠ P2) return 0;
14    return 1;
15 }
```

Axioms PL2 to PL5 are checked in much the same way; implementation of functions isPL2 to isPL5 can now be left as an exercise. Axiom PL6 however raises some other concerns: it is necessary to check whether a substitution $[x/t]$ is acceptable in an expression P and, if so, check whether another expression is of the form $P[x/t]$ or not.

In order to recognise a valid instance of PL6 we need to be able to decide, given Gödel numbers `x` and `y` of expressions and `v` of a symbol, whether the expression **y** is the result of an acceptable substitution of **any** term for `v` in the expression **x** or not. Referring back to the original definition of the function P, we see that, for all x and y, $P(x, y) \geq x$ and $P(x, y) \geq y$ and that P is strictly increasing in both variables. It follows that the Gödel number of a string is greater than the Gödel number of any of its substrings and greater than the Gödel number of any of its characters. So, in order to perform the test just described, it is enough to test for substitutions `[v/t]` for all t of Gödel number less than that of **y**. This is obviously very inefficient, but we are not interested in efficiency here, and it is simple.

```
1  anySubst(x,v,y) {
2     t:= 1;
3     while (t ≤ y) {
4        if (isTerm(t) &
5           Acceptable(x,v,t) &
6           y=SubstInExpression(x,v,t)) return 1;
7        t := t + 1; \\
8        }
9  return 0;
10 }
```

Now we can test a string to see if it is an instance of Axiom PL6. The function isPL6(**x**)

returns 1 if string **x** is an instance of Axiom PL6, 0 otherwise. This axiom in its formal form is $((\forall x P) \Rightarrow Q)$, where P is an expression, Q is the result of substituting term **t** for variable v in P and this substitution is acceptable.

```
isPL6(x; p,ch,y,z) {
   if (not isExpression(x) return 0;

   ch := ent(x,1);
   if (not isOpenSym(ch)) return 0;
   ch := ent(x,2);
   if (not isOpenSym(ch)) return 0;
   ch := ent(x,3);
   if (not isForAllSym(ch)) return 0;
   v := ent(x,4);
   if (not isVarSym(v)) return 0;
   y := GetSubexpression(x,5);
   p := skipSubexpression(x,5);
   ch := ent(x,p);
   if (not isCloseSym(ch)) return 0;
   p := p+1; ch := ent(x,p);
   if (not isImpSym(ch)) return 0;
   p := p+1;
   z := GetSubexpression(x,p);
   p := skipSubexpression(x,p);
   ch := ent(x,p);
   if (not isCloseSym(x,p)) return 0;
   if (p ≠ len(x)) return 0;

   if (not anySubst(y,z,v)) return 0;

   else return 1;
}
```

Now, of course, checking whether an string is an axiom of **PL** is trivial:

```
isAxiomOfPL(x) {
   if (isPL1(x) or isPL2(x) or isPL3(x) or
      isPL4(x) or isPL5(x) or isPL6(x)) return 1;
   else return 0;
}
```

▶ 9.A.10

B.7 Recognising proofs (for 9.A.10)

We want to be able to recognise a proof in an axiomatisable theory. The theory may have axioms other than the six of **PL**, but we are assuming that the set of all axioms is recursive. That means we may assume the existence of a function isAxiom(**x**) which returns 1 for true if the string **x** is an axiom of the system and 0 for false otherwise.

Consider first recognising Modus Ponens. This is fairly simple. Given three expressions $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$, we wish to recognise when they are of the form P, $(P \Rightarrow Q)$ and Q respectively; that amounts to simply recognising when $\mathbf{y}$ is of the form $(\mathbf{x} \Rightarrow \mathbf{z})$. Assuming that we have already checked that $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are valid expressions,

```
1  isModusPonens(x,y,z; p,ch) {
2     ch := ent(y,1);
3     if (not isOpener(ch)) return 0;
4     if (not isSubexpression(y,2)) return 0;
5     if (x ≠ GetSubexpression(y,2)) return 0;
6     p := SkipSubexpression(y,p);
7     ch := ent(y,p);
8     if (not isImpSym(ch)) return 0;
9     p := p+1;
10    if (not isSubexpression(y,p)) return 0;
11    if (z ≠ GetSubexpression(y,p)) return 0;
12    p := SkipSubexpression(y,p);
13    ch := ent(y,p);
14    if (not isCloseSym(ch)) return 0;
15    if (p ≠ len(y)) return 0;
16    else return 1;
17 }
```

For Universal Generalisation we must recognise when two strings **x** and **y** are of the form P and $(\forall v\ P)$, that is, when **y** is of the form $(\forall v\ \mathbf{x})$. This can now safely be left as an exercise.

Before we define `isProof` we need to define the function `isValidStep`. Given a (sequence) Gödel number `S` of a sequence of expressions and a number `n` less than or equal to its length, `isValidStep(S,n)` will return 1 for true if the $\mathtt{n}^{\text{th}}$ step in the sequence is valid proof-step, and 0 for false otherwise.

```
1  isValidStep(S,n; x,i,j) {
2     x := ent(S,n);
3     if (isAxiom(x)) return 1;
4     i := 1; j := 1;
5     while (i < n) {
6        while(j < n) {
7           if (isModusPonens(ent(S,i), ent(S,j), x)) return 1;
8        }
9        if (isUniversalGeneralisation(ent(S,i),x)) return 1;
10    }
11    return 0;
12 }
```

Now the function `isProof` is easy. Given a number `S`, `isProof(S)` will return 1 for true if it is the (sequence) Gödel number of a valid proof and 0 for false otherwise.

```
1  isProof(S; length,stepNum,x) {
2     length := len(S);
```

The number of steps in the proof

```
3     stepNum := 1;
4     while (stepNum < length) {                    Loop through the steps of the proof
5        x := ent(S,stepNum);                       The current step
6        if( not isExpression(x)) return 0;         Check it out
7        if( not isValidStep(S,stepNum)) return 0;
8        stepNum := stepNum + 1;
9     }
10    return 1;
11 }
```

B.8 The "Proof of" relation (for 9.A.11)

▶ 9.A.11

We want to define a partial recursive function `ProofOf(S,x)` which returns value 1 when `S` is the Gödel number of a proof of the expression with Gödel number `x` and returns 0 otherwise. This is now easy.

```
1  ProofOf(S,x) {
2     if (not isProof(S)) return 0;
3     if (x ≠ ent(S,len(S))) return 0;
4     else return 1;
5  }
```

B.9 Enumerating theorems (for 9.A.12)

▶ 9.A.12

We want to define a partial recursive function `theoremNumber` which enumerates all theorems, that is, whose range is the set of the Gödel numbers of all theorems. This is now simple.

```
1  theoremNumber(S) {
2     if (isProof(S)) return ent(S,len(S));
3  }
```

Here, if **S** is the Gödel number of a valid proof, the function returns its last entry, which is the theorem it proves; otherwise it does not return any value. It is not hard to make this a recursive function if you prefer. One way is to choose the Gödel number `n0` of your favourite simple theorem, say $P \Rightarrow P$, and redefine the function thus:

```
1  theoremNumber(S) {
2     if (isProof(S)) return ent(S,len(S));
3     else return n0;
4  }
```

Programs and routines, as we have defined them so far, simply return a single number. But suppose we had something like a "`print`" statement in our language. Then we could set our machine to print out the Gödel numbers of all theorems thus:

```
AllTheorems(;S) {
   S := 0;
   while (1) {
      if (isProof(S)) print(ent(S,len(S)));
      S :=S+1;
   }
}
```

Now all you have to do is decode the number `ent(S,len(S))` back into its actual string of characters (not hard!) and you have a program which will type out every theorem of mathematics. Just sit back and wait for your favourite theorem to turn up.

▶ 9.C.1

B.10 The W and V functions (for 9.C.1)

In order to compute this function we will need to know how to convert an ordinary number ξ into the Gödel number of the string $\bar{\xi}$ which represents it. For example, if ξ is 2, then $\bar{\xi}$ is the string $s(s(\bar{0}))$ of length 7. The algorithm will use the following constants

$$\begin{aligned}\text{The Gödel number of } \bar{0} \text{ as a string of length 1 is } P(0,6) &= 21\,,\\ \text{The Gödel number of) as a string of length 1 is } P(0,4) &= 10\,,\\ \text{The Gödel number of the string } s(\text{ of length 2 is } P(1,P(7,3)) &= 2017\,.\end{aligned}$$

and we have a recursive algorithm

```
NumberToString(n; x) {
   if (n=0) return 21;
   else {
      x := NumberToString(n-1);
      return Concat(Concat(2017,x),10);
   }
}
```

Now for the W function.

```
W(xi,eta; x,v,xix) {

   if (not isExpression(xi)) return 0;

   x := 0;
   v := 11;
```

```
5     while (v<xi) {
6        if (FreeInExpression(v,xi) {
7           if (x=0) x:=v;
8           else if (v ≠ x) return 0;
9        }
10       if (x=0) return 0;
11    }

12    xix := SubstInExpression(xi,x,NumberToString(xi));

13    if (not ProofOf(eta,xix)) return 0;

14    return 1;
15 }
```

For the V function,we use the constant

The Gödel number of the string ($\neg$ of length 2 is $P(1, P(3, 0)) = 56$.

Now modify the algorithm for W by replacing Line 17 by

```
17    nxix := Concat(Concat(56,xix),10);
18    if (not ProofOf(eta,nxix)) return 0;
```

D. TURING MACHINES

A General description

To my mind there are two important things about Turing machines.

Firstly, they were invented as a thought experiment by Alan Turing in 1937 and (with perhaps a slight challenge from Babbitt's engines) constituted the first definition of a stored-program computer. His ideas were taken up and expanded not long after by John Von Neumann leading to the first working such computers being built in the late 1940s.

But more importantly for our interests here, one can make a case for the assertion that a Turing machine is an idealisation which incorporates the idea of *any algorithm whatsoever*. (This is what Turing had in mind when he invented them.) Let me try and justify this assertion.

Consider an algorithm; it must be a cut and dried one such as the ones we discuss in Chapter **8**, so let us take addition of numbers of several digits in decimal notation as a familiar example, say

► Ch.8

$$\begin{array}{rr} & 4\,7\,3 \\ + & 2\,5\,6 \\ \hline \end{array}$$

We all (I hope) know how to do this sum. We start with the rightmost column, think "3 plus 6 is 9", write 9 at the bottom, move on to the next column and so on.

Let us analyse this process down to its very simplest components, the idea being to eventually see that any algorithm at all consists of similar components. We also want to see that, in order to perform such an algorithm, one only needs to know a finite number of these elementary "instructions" and also one only needs to keep a finite amount of information in one's mind to know where one is up to and what to do next.

In outline, what we do is

> Start with the rightmost column;
> repeat:
> > Deal with the column we are looking at
> > then move to the next column to the left
>
> until no more columns.

So what does "deal with a column" involve?

> Look at the top digit;
> Remember it and move down to the digit in the second row (of that column);
> Move down to the third row and write the sum of those two digits.

To calculate the sum of two *digits* does not require an algorithm: all it needs is for us to have memorised the addition table for decimal digits, a finite amount of information, which we are all taught in school (or could have been).

So what we do at any point depends upon what the symbol we are looking at is and what state our head is in at the time. To illustrate, let us follow the first few steps. Here I will show which symbol is being looked at by writing it boldface. The boxes in the bottom row represent empty spaces to be filled as we go.

$$\begin{array}{rr} & 4\ 7\ \mathbf{3} \\ + & 2\ 5\ 6 \\ \hline & \square\ \square\ \square \end{array}$$ Looking at first row digit;

Remember the 3 and move down one row.

$$\begin{array}{rr} & 4\ 7\ 3 \\ + & 2\ 5\ \mathbf{6} \\ \hline & \square\ \square\ \square \end{array}$$ Looking at second row, have seen a 3;

Remember the 6 also and move down one row.

$$\begin{array}{rr} & 4\ 7\ 3 \\ + & 2\ 5\ 6 \\ \hline & \square\ \square\ \boldsymbol{\square} \end{array}$$ Looking at third row, have seen a 3 and a 6;

Write 9, move to top of next column to left.

$$\begin{array}{rr} & 4\ \mathbf{7}\ 3 \\ + & 2\ 5\ 6 \\ \hline & \square\ \square\ 9 \end{array}$$ Looking at first row digit;

Remember the 7 and move down one row.

AND SO ON.

There is only a finite number of head-states needed here; as a first approximation:

> "Looking at first row digit";
>
> "Looking at second row digit, have seen an x", where x is one of the digits $0 \ldots 9$ (ten such states);
>
> "Looking at third row space, have seen an x and a y", where x and y are each one of the digits $0 \ldots 9$ (one hundred such states).

However, we need more states than that because I haven't taken into account carries and what to do if one of the numbers has more digits than the other. But it is fairly obvious what adjustments we need to make. The possible states are:

"Looking at first row digit, carrying z", where z is 0 or 1 (two such states);

"Looking at second row digit, have seen an x in the first row and carrying z", where x is a blank or a digit ($11 \times 2 = 22$ such states);

"Looking at third row space, have seen an x and a y and carrying z", where x and y are each either a blank or a digit ($11 \times 11 \times 2 = 242$ such states).

What we do next at any stage is a function of both what our current head-state is and what we are looking at; and what we do consists of changing our head-state appropriately, first perhaps writing something and perhaps making some movement (up, down, left or right). So, for instance, if in head-state

Looking at third row space, having seen 7 and 5 and carrying 0

and seeing

blank

write 2, move to top of next column to the left and change head-state to

Looking at first row digit, carrying 1.

While we are reducing everything to such a minimum, we should analyse "Move to top of next column to the left" down to its individual steps. Add a couple more head-states:

"Looking at second row digit, moving on, carrying z";

"Looking at first row digit, moving on, carrying z".

The behaviour just described above resolves itself into: Start in head-state

Looking at third row space, having seen 7 and 5 and carrying 0

and seeing

blank

write 2, move up one space (to the second row) and change head-state to

Looking at second row digit, moving on, carrying 1.

Now we are seeing 5 again, but ignore it. Write nothing, move up one space (to first row) and change head-state to

Looking at first row digit, moving on, carrying 1.

Now we are seeing 7 again, but ignore it too. Write nothing, move left one space (to first row of next column) and change head-state to

Looking at first row digit, carrying 1.

This is a pretty good description of what we actually do. The entire algorithm can be described by a list of what to do next under each circumstance that may arise:

In state	Seeing	Write	Move	Change to state
1st row, carrying 0	□	nothing	down	2nd row, seen □, carrying 0
1st row, carrying 0	0	nothing	down	2nd row, seen 0, carrying 0
1st row, carrying 0	1	nothing	down	2nd row, seen 1, carrying 0
⋮	⋮	⋮	⋮	⋮
3rd row, carrying 0, seen 7 and 5	any	2	up	2nd row moving on, carrying 1
⋮	⋮	⋮	⋮	⋮
1st row, carrying 1, moving on	any	nothing	left	1st row, carrying 1

(Here of course the entry "any" means that we take the same action whatever we are seeing. Such a line is shorthand for eleven actual lines.)

We know when we are finished by reaching a column with both entries empty. In the middle of the table above there will be two lines which take care of this:

In state	Seeing	Write	Move	Change to state
⋮	⋮	⋮	⋮	⋮
3rd row, carrying 0, seen □ and □	any	nothing	no	Finished
3rd row, carrying 1, seen □ and □	any	1	no	Finished
⋮	⋮	⋮	⋮	⋮

Finally, to be extra careful, we should specify that when the algorithm reaches the "Finished" state it remains there: we need lines of the form

In state	Seeing	Write	Move	Change to state
⋮	⋮	⋮	⋮	⋮
Finished	any	nothing	no	Finished
⋮	⋮	⋮	⋮	⋮

We an describe this whole thing mathematically as a function. Let us write

States	for the set of states,
Alphabet	for the set of symbols (the digits plus blank),
Moves	for the set of possible moves (no, up, down, left, right).

Then our algorithm is defined by a function

$$\text{States}\times\text{Alphabet} \;\to\; \text{Alphabet}\times\text{Moves}\times\text{States}$$

All these sets are finite.

Note well that this algorithm works for two numbers of any size whatsoever, without changing the size of the sets above (or, if you like, the length of the table).

So far so good. Now let us consider the algorithm for multiplying two such numbers. Here

is what it looks like when finished.

$$\begin{array}{rr} & 4\,7\,3 \\ \times & 2\,5\,6 \\ \hline & 2\,8\,3\,8 \\ & 2\,3\,6\,5 \\ & 9\,4\,6 \\ \hline & 1\,2\,1\,0\,8\,8 \end{array}$$

We see that there is a sub-algorithm involved here: to add up a column of figures such as

$$\begin{array}{rr} & 2\,8\,3\,8 \\ & 2\,3\,6\,5 \\ + & 9\,4\,6 \\ \hline \end{array}$$

Again, we know how we do this in practice: we run down each column of digits, keeping a mental tally of the sum, writing down the last digit of our sum when we get to the bottom and retaining the others as a carry. This is all very well, but if the number of rows in our array is arbitrarily large, we must be prepared to retain an arbitrarily large working sum in our heads as we go down the column. This corresponds to an arbitrarily large number of different head-states and the specification of the algorithm ceases to be finite.

Indeed, we want an algorithm that will work for numbers of any size whatsoever. It can take as long as it likes and use as much paper as it likes, but the number of symbols, states and moves must be finite. The standard algorithm, the one we usually use, breaks down if, say, the numbers have 10^{1000} digits, because there is no way one can remember a working sum as large as that; our finite brains run out of memory.

(By the way, we need the algorithm to have an entirely finite specification. Otherwise we could simply say, "Memorise the entire infinite multiplication table $\mathbb{N}\times\mathbb{N} \to \mathbb{N}$ and just look it up". In this way any algorithm becomes completely trivial.)

One way of dealing with additions with many rows and columns is to calculate cumulative sums as one goes down, in the manner of a bank statement, for instance

1 5 7 3	1 5 7 3
2 6 8 1	4 2 5 4
4 5 3	4 7 0 7
2 1 9 0	6 8 9 7
8 0 0 1	1 4 8 9 8
3 2	1 4 9 3 0

Here the left hand column is the set of numbers to be summed. The right hand column develops the cumulative sums; each entry in this column is the sum of the number directly above it and the one on its left. The final figure is the bottom right hand one.

This algorithm (which is in fact the one usually used when large numbers of figures must be added by hand) can be described in a finitary way using a finite state changing function such as the one above. There is one problem that needs addressing before this can be done: the algorithm involves a fair amount of jumping around. But one cannot remember where to go next in forms such as "the 10th digit of the 37th entry in the accumulation column"

because being able to remember the counting numbers here would imply that the number of head-states be infinite. (Think again of numbers larger than 10^{1000} here.)

This is not difficult to work around: just use tags. For instance, here we are, part way through, working on the hundreds digit in the 4th line, using tags (represented by bars) to keep track of the various relevant places:

1 5 7 3	1 5 7 3
2 6 8 1	4 2 5 4
4 5 3	4 $\bar{7}$ 0 7
2 $\bar{1}$ 9 0	$\bar{\square}$ 9 7
8 0 0 1	
3 2	

In doing this I have enlarged the alphabet of course: I now have a blank, the ten digits and ten tagged digits. There is nothing to stop us using a variety of tags, as many as we like (so long as finitely many).

It is now a fairly straightforward, if tedious, exercise to write this algorithm out as a finite next-state table like the one above.

The point of all this is, of course, to convince ourselves that this kind of process can be applied to *any algorithm whatsoever* to reduce it to a finite next-state table description.

Before going on to the standard definition of a Turing machine, we ask two questions.

Firstly, *noticing that our algorithms above are performed in a two-dimensional array, is there anything special about two dimensions here*?

Not really. Two dimensional arrays for algorithms, as actually performed by hand, are very common presumably because most paper is two dimensional. Three dimensions or more would be very convenient for some algorithms if only we had some convenient way of reading and writing this way. It has been proved that any number of dimensions (except 0) is sufficient for all algorithms. For simplicity, Turing machines use one-dimensional paper (known as a tape).

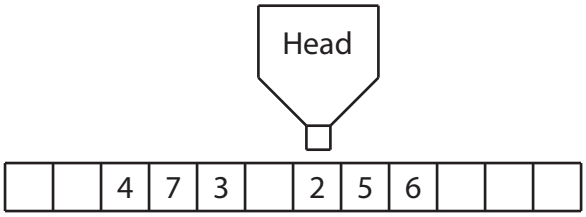
Incidentally, you will have noticed that it is assumed that the symbols are written down in a neat array, with rows and columns, as though we are using squared paper. This is pretty well essential to keep track of moving about. Our one-dimensional tape will be an infinite sequence of squares, each capable of holding a single symbol.

Secondly, *what difference does the size of the alphabet make*?

None, provided that it has at least two symbols. The algorithms that can be computed with an alphabet of size n ($n \geq 2$) are the same as those that can be computed with an alphabet of any other size (≥ 2 again).

B Details

Visualise a Turing machine as a robot looking at a tape. The tape is made up of squares ("cells") and is infinitely long in both directions — or at least potentially infinitely long, in the sense that the robot can extend either end indefinitely as needed. The robot has a "read-write head" that at any time is looking at a particular cell of the tape; it can move the tape backwards and forwards, one cell at a time and it can write symbols of its alphabet into the cells of the tape (erasing whatever was there before).



Every Turing machine has the same set of *moves*, $M = \{L, O, R\}$. As far as moving goes, it is more convenient to think of the robot as moving its head back and forth over the tape than as standing still and moving the tape. So L means "move left one cell'. R means "move right one cell" and O means "don't move".

Among the symbols which can be written on the tape is one distinguished one, the *blank*, which we will write as $\Box$. At any time, all but a finite number of the squares on the tape contain this blank symbol. (Think of the tape as being all blank before it is written on.)

Among the states the machine can be in are two distinguished ones, the *initial* state I, which is the state we set the machine in to start it going, and the *final* state F, which is the state which signals it has finished its calculations. Once the machine reaches state F, it stops: it makes no further changes of state, no further moves and does no more writing.

A particular Turing machine is defined by the following things:

(a) A finite set S, its set of *states*, containing two special elements, I and F, the *initial* and *final* states. For convenience we write S^- for the set $S \setminus \{F\}$.

(b) A finite set A, its *alphabet*, containing one special element, $\Box$, the *blank*.

(c) A finite set M of *moves*, discussed above.

(d) A *write* function $\omega : S^- \times A \to A$.

(e) A *move* function $\mu : S^- \times A \to M$.

(f) The *next state function* $\sigma : S^- \times A \to S$.

This is to be interpreted as follows. We start the machine with the tape in its initial condition, containing whatever input we wish the machine to operate on. The tape will be

mostly blank. Thus the tape will be defined as a doubly-infinite sequence

$$\ldots, a_{-2}\,,\, a_{-1}\,,\, a_0\,,\, a_1\,,\, a_2\,,\, \ldots$$

all but a finite number of whose elements are blank.

The machine's R/W head will initially be positioned over (looking at) a particular member of this sequence, let us say a_0. We can picture this thus:

$$\ldots, a_{-2}\,,\, a_{-1}\,,\, \overset{\nabla}{a_0}\,,\, a_1\,,\, a_2\,,\, \ldots$$

And, of course, we can dispense with the dots at either end. The picture which started this section could be described thus:

$$4\ \ 7\ \ 3\ \ \square\ \overset{\nabla}{2}\ \ 5\ \ 6$$

and if we wanted to show that the machine was in state S while looking at the cell with 2 in it, we could picture it thus:

$$\begin{array}{cccccccc} & & & & S & & & \\ & & & & \nabla & & & \\ 4 & 7 & 3 & \square & 2 & 5 & 6 \end{array}$$

Such a picture is a description of the machine at any particular instant. One can think of it as a convenient picture of the *instantaneous description* or *ID* of the machine, which consists of three things

(a) The tape expression E, which is a doubly-infinite sequence of symbols the alphabet,

(b) The head position p, which is the index of the entry in the sequence it is looking at and

(c) The state s the machine is in at the time.

So we can define an ID of the machine to be the triple (s, p, E). If we write

$$E = (\ldots\,,\, e_{-2}\,,\, e_{-1}\,,\, e_0\,,\, e_1\,,\, e_2\,,\, \ldots)$$

then we can describe the *transition function*, which describes what the machine does next, that is, the ID it goes to next. This is prescribed by the three functions above.

The machine is in state s and is looking at cell e_p, so

- It changes e_p to the value $\omega(s, e_p)$, then
- It moves left or right (decrements or increments p) according to $\mu(s, e_p)$ (the old e_p), then
- It changes its state to $\sigma(s, e_p)$ (again the old e_p).

If T is the Turing machine and α and β are the before and after states, we can denote this by $T : \alpha \to \beta$.

A *computation* by the Turing machine T is a sequence of IDs

$$\alpha_0\,,\, \alpha_1\,,\, \alpha_2\,,\, \ldots$$

such that $T : \alpha_0 \to \alpha_1 \to \alpha_2 \to \ldots$.

A final state F, if it occurs, terminates the computation. Thus we have two possibilities:

An ID occurs which is a final state one, in which case only one such ID occurs in the sequence and that is the last one. In this case the computation is said to *halt*.

Otherwise there is no ID in the computation which is a final state one, in which case the computation sequence is infinite. In this case the computation does not halt.

B.1 Number codes

We will look at some examples. It really doesn't matter in the long run what code we use for natural numbers, and we are not particularly interested in human-readability, so let us use *base-1 notation*. In this notation, the numbers are encoded thus:

Number	Code
0	(blank)
1	1
2	11
3	111

and so on: n is encoded as a string of n 1's.

We encode n-tuples thus:

$$(a_1, a_2, \ldots, a_n) \quad \text{is encoded as} \quad A_1 \square A_d \square \ldots \square A_n$$
$$\text{where} \quad A_i \text{ is the encoding above for } a_i, \text{ for } i = 1, 2, \ldots, n$$

for example

$$(\,2\,,\,0\,,\,3\,,\,1\,,\,0\,) \quad \text{is encoded as} \quad 1\ \ 1\ \ \square\ \ \square\ \ 1\ \ 1\ \ 1\ \ \square\ \ 1\ \ \square$$

C Examples of Turing machines

I here give a number of examples of Turing machines, building up from simpler ones to more complicate ones. The aim is to show that the basic functions and operations defining partial recursive functions (as given in Definition **8**.A.2) can all be computed by Turing machines. This shows that any partial recursive function can be computed by a suitable Turing machine.

▶ 8.A.2

We will adopt the convention that a Turing machine computes the function $f : \mathbb{N}^n \to \mathbb{N}$ if it computes

$$\overset{\nabla}{A_1} \quad A_2 \quad \ldots \quad A_n \quad \to \quad \overset{\nabla}{B} \tag{–1}$$

whenever $f(a_1, a_2, \ldots, a_n) = b$ and $A_1, A_2, \ldots, A_n$ and B are the codes for $a_1, a_2, \ldots, a_n$ and b.

We will of course be interested in substituting one computation into another in various ways. This will mean that we will be interested in *embedded* computations, where the cells of interest are surrounded codes for n-tuples and remain thus surrounded throughout. For example, an embedded computation of the function f above would compute

$$\overset{\nabla}{X} \quad A_1 \quad A_2 \quad \ldots \quad A_n \quad Y \quad \to \quad \overset{\nabla}{X} \quad B \quad Y \tag{–2}$$

where X and Y are codes for a k-tuple and an l-tuple. In embedded computations, the dimensions k and l will always be known: a single machine does not have to be designed for variable values of these dimensions. In other words, for different values of k and l it will be sufficient for our purposes to build different machines.

We will call (–2) the $(k : l)$-*embedded form* of (–1) .

Note also that, in order to facilitate building more complicated machines from simpler ones, we usually adopt the convention that the machine starts and finishes looking at the leftmost relevant cell.

C.1 A Turing machine to step right across a single number code

The machine assumes that it starts looking at the first digit of a number. It steps to the right across that number and ends up looking at the first digit of the next number.

$$\overset{\nabla}{A} \; \square \; B \quad \to \quad A \; \square \; \overset{\nabla}{B} \; .$$

This is easy:

In state	Seeing	Write	Move	To state	Comment
init	1	–	R	init	Step right to first blank.
init	$\square$	–	R	fin	Step right to start of B.
fin	any	–	O	fin	Final.

C.2 A Turing machine to step left across a single number code

The machine assumes that it starts looking at the first digit of a number. It steps to the left across that number and ends up looking at the first digit of the previous number.

$$\square\ A\ \square\ \overset{\triangledown}{B} \quad \rightarrow \quad \square\ \overset{\triangledown}{A}\ \square\ B\ .$$

Note that simply changing R to L in the above recipe does not do this.

In state	Seeing	Write	Move	To state	Comment
init	any	–	L	s_1	$\square\ A\ \overset{\triangledown}{\square}\ B$
s_1	any	–	L	s_2	$\square\ \overset{\triangledown}{A}\ \square\ B$
s_2	1	–	L	s_2	$\overset{\triangledown}{\square}\ A\ \square\ B$
s_2	0	–	R	fin	$\square\ \overset{\triangledown}{A}\ \square\ B$
fin	any	–	O	fin	Final.

C.3 A Turing machine to compute $(a, b) \rightarrow a + b$, first try

This machine computes

$$\overset{\triangledown}{A}\ \square\ B \quad \rightarrow \quad \overset{\triangledown}{C}\ \square\ ,$$

where A and B are single number codes and C is the code for their sum. If we show what the computation might be embedded in,

$$X\ \square\ \overset{\triangledown}{A}\ \square\ B\ \square\ Y \quad \rightarrow \quad X\ \square\ \overset{\triangledown}{C}\ \square\ \square\ Y\ ,$$

where X and Y are the codes for an k-tuple and an l-tuple.

With the code we are using, addition is ridiculously easy: all we need do is concatenate the codes. For example, to add $2 + 3$, it computes:

$$\overset{\triangledown}{1}\ 1\ \square\ 1\ 1\ 1\ \square \quad \rightarrow \quad \overset{\triangledown}{1}\ 1\ 1\ 1\ 1\ \square\ \square$$

To compute the sum, all it has to do is replace the first blank by a 1 and the last 1 by a blank. Note that it must be able to cope with zeros:

$$0 + b: \quad \overset{\triangledown}{\square}\ 1\ 1\ 1\ \square$$
$$a + 0: \quad \overset{\triangledown}{1}\ 1\ \square\ \square$$

or even

$$0 + 0: \quad \overset{\triangledown}{\square}\ \square$$

Here I have written init and fin for the initial and final states, s_1, s_2, s_3 for the intermediate ones.

In state	Seeing	Write	Move	To state	Comment
init	1	–	R	init	Step right across a.
init	□	1	R	s_1	Found first blank; put 1 there.
s_1	1	–	R	s_1	Step right across b.
s_1	□	–	L	s_2	Found second blank; back up one.
s_2	1	□	O	s_3	Replace last 1 (if any) by □.
s_2	□	–	O	s_3	Maybe there isn't a last 1.
s_3	1	–	L	s_3	Step left across the sum.
s_3	□	–	R	fin	Move right one cell.
fin	any	–	O	fin	Final.

Note that this computation is oversimplified, and not what we really want. Firstly, for an embedded computation, we need the machine to start and end looking at the leftmost digit of X and, secondly, we must get rid of that extra blank to the right of the sum C. What we need is

$$\overset{\triangledown}{X}\ \square\ A\ \square\ B\ \square\ Y\ \ \rightarrow\ \ \overset{\triangledown}{X}\ \square\ C\ \square\ Y\ ,$$

and first we need a few simpler operations to do this.

C.4 Concatenation of computations

Given two Turing machines T_1 and T_2, we may wish to *concatenate* them, that is, build a machine which does what T_1 does followed by what T_2 does.

For example, concatenating two copies of the machine just defined to step right across a single number code, we would produce a machine that stepped right across two number codes.

The computations of two Turing machines T_1 and T_2 may be composed quite easily by using the following recipe: Write down the specification of T_1 followed by that of T_2, *then*

1. Change the state set of T_2 if necessary so that it has no states in common with T_1. In particular, the initial state of T_2 is changed to a non-initial state, J say.

2. Remove the last line of the T_1 specification, which says that in the state fin it simply does nothing.

3. Change any earlier occurrence of the state fin in T_1 to J.

So the machine to step right across two number codes would look like this

$$\overset{\triangledown}{A}\ \square\ B\ \square\ C\ \ \rightarrow\ \ A\ \square\ B\ \square\ \overset{\triangledown}{C}\ .$$

In state	Seeing	Write	Move	To state	Comment
init	1	–	R	init	Step right to first blank.
init	□	–	R	J	Step right to start of B.
J	1	–	R	J	Step right to first blank.
J	□	–	R	fin	Step right to start of B.
fin	any	–	O	fin	Final.

As a more interesting example we construct a Turing machine to add three numbers. It should perform

$$\overset{\nabla}{A}\ \square\ B\ \square\ C\ \square \quad \rightarrow \quad \overset{\nabla}{A}\ B\ C\ \square\ \square\ \square$$

This can be done by concatenating operations we know already:

$$\begin{aligned} \text{Step right across a single number} &\rightarrow A\ \square\ \overset{\nabla}{B}\ \square\ C\ \square \\ \text{Add} &\rightarrow A\ \square\ \overset{\nabla}{B\,C}\ \square\ \square \\ \text{Step left across a single number} &\rightarrow \overset{\nabla}{A}\ \square\ BC\ \square\ \square \\ \text{Add} &\rightarrow \overset{\nabla}{A\,BC}\ \square\ \square\ \square \end{aligned}$$

So, applying the recipe,

In state	Seeing	Write	Move	To state	Comment
init	1	–	R	init	Step right across single number
init	□	–	R	s_0	
s_0	1	–	R	s_0	
s_0	□	1	R	s_1	
s_1	1	–	R	s_1	
s_1	□	–	L	s_2	Add
s_2	1	□	O	s_3	
s_2	□	–	O	s_3	
s_3	1	–	L	s_3	
s_3	□	–	R	t_0	
t_0	any	–	L	t_1	
t_1	any	–	L	t_2	Step left across a single number
t_2	1	–	L	t_2	
t_2	□	–	R	u_0	
u_0	1	–	R	u_0	
u_0	□	1	R	u_1	
u_1	1	–	R	u_1	
u_1	□	–	L	u_2	
u_2	1	□	O	u_3	Add
u_2	□	–	O	u_3	
u_3	1	–	L	u_3	
u_3	□	–	R	fin	
fin	any	–	O	fin	

C.5 A Turing machines to step right or left across an n-tuple

Consider stepping right across an n-tuple code

$$\overset{\nabla}{A_1}\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ B\ \ \rightarrow\ \ A_1\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ \overset{\nabla}{B}\ .$$

Here the machine starts looking at the first digit of the first number code in the n-tuple code $A_1\ \square\ A_2\ \square\ \ldots\ \square\ A_n$. It steps across the n-tuple, coming to rest looking at the first digit of whatever number follows it.

The reason for going that extra step is to end up at the beginning of whatever number (here written B) comes after. This is important for the uses we will put it to. The machine is designed for a particular value of n.

To build the machine is easy: simply concatenate n copies of the machine to step right across a single number code, discussed above.

In the same way, concatenating n copies of the machine which steps left across a number code creates a machine which steps left across an n-tuple:

$$A_1\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ \overset{\nabla}{B}\ \ \rightarrow\ \ \overset{\nabla}{A_1}\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ B\ .$$

C.6 A machine to shift a number code one space to the left

This machine computes

$$\square\ \overset{\nabla}{\square}\ A\ \square\ \ \rightarrow\ \ \square\ \overset{\nabla}{A}\ \square\ \square$$

where A is the code for a single number. Note that this will only be used when there is an extra blank to the left, as shown. The machine needs to start by stepping one cell to the right to check if A is empty: if it is, it can stop right there.

In state	Seeing	Write	Move	To state	Result
init	$\square$	–	R	s_1	$\square\ \square\ \overset{\nabla}{A}\ \square$
s_1	$\square$	–	L	fin	A is empty; finish.
s_1	1	–	L	s_2	$\square\ \overset{\nabla}{\square}\ A\ \square$
s_2	$\square$	1	R	s_3	$\square\ 1\ \overset{\nabla}{A}\ \square$
s_3	1	–	R	s_3	$\square\ 1\ A\ \overset{\nabla}{\square}$
s_3	$\square$	–	L	s_4	$\square\ 1\ \overset{\nabla}{A}\ \square = \square\ A\ \overset{\nabla}{1}\ \square$
s_4	1	$\square$	L	s_5	$\square\ \overset{\nabla}{A}\ \square\ \square$
s_5	1	–	L	s_5	$\overset{\nabla}{\square}\ A\ \square\ \square$
s_5	$\square$	–	R	fin	$\square\ \overset{\nabla}{A}\ \square\ \square$
fin	any	–	O	fin	Final.

C.7 A machine to shift an n-tuple code one space to the left

This machine computes

$$\square \overset{\triangledown}{\square} A_1 \square A_2 \square \ \ldots \ \square A_n \square \ \rightarrow \ \square \overset{\triangledown}{A_1} \square A_2 \square \ \ldots \ \square A_n \square \square$$

We simply use the previous machine multiple times, as follows:

$$\begin{array}{rcl}
\text{Shift a number one cell to left} & \rightarrow & \square \overset{\triangledown}{A_1} \square \square A_2 \square \ \ldots \ \square A_n \square \\
\text{Step right over a number and one more cell} & \rightarrow & \square A_1 \square \overset{\triangledown}{\square} A_2 \square \ \ldots \ \square A_n \square \\
\text{Shift a number one cell to left} & \rightarrow & \square A_1 \square \overset{\triangledown}{A_2} \square \square \ \ldots \ \square A_n \square \\
\vdots & & \\
\text{and so on down to} & & \\
\text{Step right over a number and one more cell} & \rightarrow & \square A_1 \square A_2 \square \ \ldots \ \square A_n \square \overset{\triangledown}{\square} \\
\text{Step left over an } (n+1)\text{-tuple} & \rightarrow & \square \overset{\triangledown}{A_1} \square A_2 \square \ \ldots \ \square A_n \square \square \\
\text{Finish} & &
\end{array}$$

C.8 Embedded addition

This machine computes

$$\overset{\triangledown}{X} \ \square \ A \ \square \ B \ \square \ Y \ \rightarrow \ \overset{\triangledown}{X} \ \square \ C \ \square \ Y$$

where X is the code for a k-tuple, Y the code for an l-tuple, A and B codes for numbers, a and b say, and C the code for the sum $a+b$. (The machine is designed for a specific m and n; in other words, for a different m and n you need a different machine.)

As already remarked, this is fairly straightforward, because it is done simply by removing the blank between A and B; the machine computes

$$\overset{\triangledown}{X} \ \square \ A \ \square \ B \ \square \ Y \ \rightarrow \ \overset{\triangledown}{X} \ \square \ A\,B \ \square \ Y$$

This can be done by composing operations we know already:

$$\begin{array}{rcl}
\text{Step right across a } k\text{-tuple} & \rightarrow & X \ \square \ \overset{\triangledown}{A} \ \square \ B \ \square \ Y \\
\text{Add ("first try" version, above)} & \rightarrow & X \ \square \ \overset{\triangledown}{C} \ \square \ \square \ Y \\
\text{Step right across one number} & \rightarrow & X \ \square \ C \ \overset{\triangledown}{\square} \ \square \ Y \\
\text{Step right one cell} & \rightarrow & X \ \square \ C \ \square \ \overset{\triangledown}{\square} \ Y \\
\text{Shift } l\text{-tuple one space left} & \rightarrow & X \ \square \ C \ \square \ \overset{\triangledown}{Y} \\
\text{Shift left across } (k+1)\text{-tuple} & \rightarrow & \overset{\triangledown}{X} \ \square \ C \ \square \ Y
\end{array}$$

C.9 Embedded deletion of a number

This machine computes

$$\overset{\triangledown}{X}\ \square\ A\ \square\ Y \quad \rightarrow \quad \overset{\triangledown}{X}\ \square\ Y\ .$$

Note that, as A is deleted digit by digit, we must shift Y to the left, so that extra space is not created.

Move right across k-tuple $\rightarrow$ $X\ \square\ \overset{\triangledown}{A}\ \square\ Y$

(1) If looking at blank, go to (2)

Else write a blank $\rightarrow$ $X\ \square\ \overset{\triangledown}{\square}\ A'\ \square\ Y$

Shift $(l+1)$-tuple one place to the left $\rightarrow$ $X\ \square\ \overset{\triangledown}{A'}\ \square\ Y$

Go to (1)

(2) If we get here, we have $X\ \overset{\triangledown}{\square}\ Y$ so

Move left across k-tuple $\overset{\triangledown}{X}\ \square\ Y$

Finish

C.10 Embedded deletion of an n-tuple

This machine computes

$$\overset{\triangledown}{X}\ \square\ A_1\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ Y \quad \rightarrow \quad \overset{\triangledown}{X}\ \square\ Y\ .$$

Easy: concatenate n copies of the last machine.

C.11 Natural subtraction

This machine computes

$$\overset{\triangledown}{X}\ \square\ A\ \square\ B\ \square\ Y \quad \rightarrow \quad \overset{\triangledown}{X}\ \square\ C\ \square\ Y$$

where A and B are codes for numbers a and b, C is code for $a \dot{-} b$ and X and Y are codes for a k-tuple and an l-tuple.

	Step right across an k-tuple	$\rightarrow$	$X\ \square\ \overset{\triangledown}{A}\ \square\ B\ \square\ Y$
(1)	If looking at a blank (i.e. a=0)		go to (2) below
	Else step right across a number	$\rightarrow$	$X\ \square\ A\ \square\ \overset{\triangledown}{B}\ \square\ Y$
	If looking at a blank (i.e. $b = 0$)		go to (3) below
	Else write a blank	$\rightarrow$	$X\ \square\ A\ \square\ \overset{\triangledown}{\square}\ B'\ \square\ Y$

(B' is B with a 1 removed)

Shift $(n+1)$-tuple left one cell	$\rightarrow$	$X\ \square\ A\ \square\ \overset{\triangledown}{B'}\ \square\ Y$
Step left one cell	$\rightarrow$	$X\ \square\ A\ \overset{\triangledown}{\square}\ B'\ \square\ Y$
Step left across one number	$\rightarrow$	$X\ \square\ \overset{\triangledown}{A}\ \square\ B'\ \square\ Y$
Write a blank	$\rightarrow$	$X\ \square\ \overset{\triangledown}{\square}\ A'\ \square\ B'\ \square\ Y$

(A' is A with a 1 removed)

Shift $(l+2)$-tuple left one cell and go to (1) above	$\rightarrow$	$X\ \square\ \overset{\triangledown}{A'}\ \square\ B'\ \square\ Y$

(2) If we get here we have $X\ \square\ \overset{\triangledown}{\square}\ B\ \square\ Y$ and should return zero.

Step right one cell	$\rightarrow$	$X\ \square\ \square\ \overset{\triangledown}{B}\ \square\ Y$
Delete number	$\rightarrow$	$X\ \square\ \square\ \overset{\triangledown}{Y}$
Step left across $(k+1)$-tuple	$\rightarrow$	$\overset{\triangledown}{X}\ \square\ \square\ Y$

Finish (the crack between the two blanks is code for 0)

(3) If we get here we have $X \ \square \ A \ \square \ \overset{\nabla}{\square} \ Y$

$$\begin{aligned}
\text{Shift } l\text{-uple one cell left} &\rightarrow X \ \square \ A \ \square \ \overset{\nabla}{Y} \\
\text{Step left across an } (k+1)\text{-tuple} &\rightarrow \overset{\nabla}{X} \ \square \ A \ \square \ Y \\
&\text{Finish}
\end{aligned}$$

C.12 Embedded machines for successor, projection functions and constant 0

These can now be safely left as exercises.

C.13 Embedded duplication of a number code

We will need this for multiplication

This machine computes

$$\overset{\nabla}{X} \ \square \ A \ \square \ Y \ \rightarrow \ \overset{\nabla}{X} \ \square \ A \ \square \ A \ \square \ Y \, .$$

$$\begin{aligned}
\text{Step right across } (k+1)\text{-tuple} &\rightarrow X \ \square \ A \ \square \ \overset{\nabla}{Y} \\
\text{Shift } l\text{-tuple one cell right} &\rightarrow X \ \square \ A \ \square \ \overset{\nabla}{\square} \ Y \\
\text{Shift } (l+1)\text{-tuple one cell right} &\rightarrow X \ \square \ A \ \square \ \overset{\nabla}{\square} \ \square \ Y \\
\text{Step two cells to left} &\rightarrow X \ \square \ \overset{\nabla}{A} \ \square \ \square \ \square \ Y \\
\text{(1) If looking at blank, go to (2), else } \ldots &
\end{aligned}$$

If we get here, A is not empty. Think of it as $A'1$ and the whole string as $X \ \square \ A' \ \overset{\nabla}{1} \ \square \ B \ \square \ B \ \square \ Y$, where B is currently empty, that is, the code for 0. We now go into a loop which successively deletes the last digit from A and adds one to each of the Bs, until A is empty.

$$\begin{aligned}
\text{Write a blank} &\rightarrow X \ \square \ A' \ \overset{\nabla}{\square} \ \square \ B \ \square \ B \ \square \ Y \\
\text{Step one cell to right} &\rightarrow X \ \square \ A' \ \square \ \overset{\nabla}{\square} \ B \ \square \ B \ \square \ Y \\
\text{Write 1} &\rightarrow X \ \square \ A' \ \square \ \overset{\nabla}{1} \ B \ \square \ B \ \square \ Y \\
\text{Step right across a number} &\rightarrow X \ \square \ A' \ \square \ 1 \ B \ \square \ \overset{\nabla}{B} \ \square \ Y \\
\text{Shift } (l+1)\text{-tuple right one cell} &\rightarrow X \ \square \ A' \ \square \ 1 \ B \ \square \ \overset{\nabla}{\square} \ B \ \square \ Y \\
\text{Write 1} &\rightarrow X \ \square \ A' \ \square \ 1 \ B \ \square \ \overset{\nabla}{1} \ B \ \square \ Y \\
\text{Step left across a number, then two more cells} &\rightarrow X \ \square \ \overset{\nabla}{A'} \ \square \ 1 \ B \ \square \ 1 \ B \ \square \ Y \\
&\text{and go to (1)}
\end{aligned}$$

When we get to here, we have $X \overset{\nabla}{\square} \square\ A\ \square\ A\ \square\ Y$, so tidy up:

$$\begin{array}{rcl}
\text{Step right one cell} & \to & X\ \square\ \overset{\nabla}{\square}\ A\ \square\ A\ \square\ Y \\
\text{Shift } (l+2)\text{-tuple one cell to left} & \to & X\ \square\ \overset{\nabla}{A}\ \square\ A\ \square\ Y \\
\text{Step left across } k\text{-tuple} & \to & \overset{\nabla}{X}\ \square\ A\ \square\ A\ \square\ Y \\
\text{finish} & &
\end{array}$$

C.14 Embedded multiplication

This machine computes

$$\overset{\nabla}{X}\ \square\ A\ \square\ B\ \square\ Y \quad \to \quad \overset{\nabla}{X}\ \square\ P\ \square\ Y\ .$$

where A and B are codes for two numbers, a and b say, and P is code for their product $p = ab$.

$$\begin{array}{rcl}
\text{Step right across } (k+2)\text{-tuple} & \to & X\ \square\ A\ \square\ B\ \square\ \overset{\nabla}{Y} \\
\text{Shift } l\text{-tuple one place right} & \to & X\ \square\ A\ \square\ B\ \square\ \overset{\nabla}{\square}\ Y \\
\text{Step left across 3-tuple} & \to & X\ \square\ \overset{\nabla}{A}\ \square\ B\ \square\ \square\ Y \\
\text{If looking at blank, go to (2)} & &
\end{array}$$

(1) If we get here, A is not empty; think of it as $1A'$; also think of the whole string as $X\ \square\ \overset{\nabla}{1}\ A'\ \square\ B\ \square\ P\ \square\ Y$, with P initially empty. The product will be developed here.

$$\begin{array}{rcl}
\text{Write blank} & \to & X\ \square\ \overset{\nabla}{\square}\ A'\ \square\ B\ \square\ P\ \square\ Y \\
\text{Shift } (l+3)\text{-tuple one cell to left} & \to & X\ \square\ \overset{\nabla}{A'}\ \square\ B\ \square\ P\ \square\ Y \\
\text{Duplicate, } (1:1+l)\text{-embedded} & \to & X\ \square\ \overset{\nabla}{A'}\ \square\ B\ \square\ B\ \square\ P\ \square\ Y \\
\text{Add, } (2:l)\text{-embedded} & \to & X\ \square\ \overset{\nabla}{A'}\ \square\ B\ \square\ B\,P\ \square\ Y \\
\text{and go to (1)} & &
\end{array}$$

(2) When we get here, we have $X \overset{\nabla}{\square} \square\ B \square\ P \square\ Y$ and P is the code for the product. Now tidy up:

$$\begin{aligned}
\text{Step right two cells} &\rightarrow X \square \square \overset{\nabla}{B} \square\ P \square\ Y \\
\text{Delete number} &\rightarrow X \square \square \overset{\nabla}{P} \square\ Y \\
\text{Step one cell to left} &\rightarrow X \square \overset{\nabla}{\square} P \square\ Y \\
\text{Shift } (l+1)\text{-tuple one cell to left} &\rightarrow X \square \overset{\nabla}{P} \square\ Y \\
\text{Step left across one number} &\rightarrow \overset{\nabla}{X} \square\ P \square\ Y \\
&\text{Finish}
\end{aligned}$$

C.15 Embedded duplication of an n-tuple code

We will need this for substitution.

First we will need a machine to duplicate the first number code in an n-tuple code to the right of the whole n-tuple, like this:

$$\overset{\nabla}{X} \square\ A_1 \square\ A_2 \square\ \ldots \square\ A_n \square\ Y \rightarrow \overset{\nabla}{X} \square\ A_1 \square\ A_2 \square\ \ldots \square\ A_n \square\ A_1 \square\ Y\,.$$

We will need a (temporary) name for this operation. Let us call it *subduplication*. In fact, we will want to be explicit about the dimensions of the various tuples involved: if X is a k-tuple and Y an l-tuple, we will call it $(k : n : l)$-*subduplication*.

The machine to compute this is very similar to the one to duplicate a number code, described above.

$$\begin{aligned}
\text{Duplicate number, } (k : n-1+l)\text{-embedded} &\rightarrow \overset{\nabla}{X} \square\ A_1 \square\ A_1 \square\ A_2 \square\ \ldots \square\ A_n \square\ Y \\
\text{Step right across } (k+n+1)\text{-tuple} &\rightarrow X \square\ A_1 \square\ A_1 \square\ A_2 \square\ \ldots \square\ A_n \square \overset{\nabla}{Y} \\
\text{Shift } n\text{-tuple one cell right} &\rightarrow X \square\ A_1 \square\ A_1 \square\ A_2 \square\ \ldots \square\ A_n \square \overset{\nabla}{\square}\ Y \\
\text{Step left across } n\text{-tuple} &\rightarrow X \square\ A_1 \square \overset{\nabla}{A_1} \square\ A_2 \square\ \ldots \square\ A_n \square \square\ Y
\end{aligned}$$

(1) If looking at blank, go to (2), else ...

If we get here, A_1 is not empty. Think of it as $1A_1'$ and the whole string as

$$X \square\ A_1 \square \overset{\nabla}{1}\ A_1' \square\ A_2 \square\ \ldots \square\ A_n \square\ B \square\ Y\,,$$

where B is currently empty, that is, the code for 0. We now go into a loop which successively

deletes the first digit from A_1 and adds one to B, until A is empty.

$$\begin{aligned}
\text{Write a blank} &\to X\ \square\ A_1\ \square\ \overset{\nabla}{\square}\ A_1'\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ B\ \square\ Y\\
\text{Shift } (n+1+l)\text{-tuple one cell to left} &\to X\ \square\ A_1\ \square\ \overset{\nabla}{A_1'}\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ B\ \square\ Y\\
\text{Step right across } n\text{-tuple} &\to X\ \square\ A_1\ \square\ A_1'\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ \overset{\nabla}{B}\ \square\ Y\\
\text{Shift } (k+1)\text{-tuple one cell to right} &\to X\ \square\ A_1\ \square\ A_1'\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ \overset{\nabla}{\square}\ B\ \square\ Y\\
\text{Write 1} &\to X\ \square\ A_1\ \square\ A_1'\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ \overset{\nabla}{1}\ B\ \square\ Y\\
\text{Step left across } n\text{-tuple} &\to X\ \square\ A_1\ \square\ \overset{\nabla}{A_1'}\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ 1\ B\ \square\ Y\\
\text{and go to (1)}&
\end{aligned}$$

When we get to here, we have

$$X\ \square\ A_1\ \square\ \overset{\nabla}{\square}\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ A_1\ \square\ Y\,,$$

so tidy up:

$$\begin{aligned}
\text{Shift } (n+k)\text{-tuple one cell to left} &\to X\ \square\ A_1\ \square\ \overset{\nabla}{A_2}\ \square\ \ldots\ \square\ A_n\ \square\ A_1\ \square\ Y\\
\text{Step left across } (k+1)\text{-tuple} &\to \overset{\nabla}{X}\ \square\ A_1\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ A_1\ \square\ Y\\
\text{finish}&
\end{aligned}$$

And now we can define the machine which duplicates an entire n-tuple ($(k:l)$-embedded). It computes

$$\overset{\nabla}{X}\ \square\ A_1\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ Y\ \to\ \overset{\nabla}{X}\ \square\ A_1\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ A_1\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ Y\,.$$

But this is now easy: starting with $\overset{\nabla}{X}\ \square\ A_1\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ Y$

$$\begin{aligned}
(k:n:l)\text{-subduplicate} &\to \overset{\nabla}{X}\ \square\ A_1\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ A_1\ \square\ Y\\
(k+1:n:l)\text{-subduplicate} &\to \overset{\nabla}{X}\ \square\ A_1\ \square\ A_2\ \square\ \ldots\ \square\ A_n\ \square\ A_1\ \square\ A_2\ \square\ Y\\
&\vdots
\end{aligned}$$

and so on, down to $(k+n-1:n:l)$-subduplicate, then finish.

C.16 Embedded substitution

Given functions $f_1, f_2, \ldots, f_n : \mathbb{N}^m \to \mathbb{N}$ and $g : \mathbb{N}^n \to \mathbb{N}$, substitution gives the function $h = g(f_1, f_2, \ldots, f_n)$. Supposing we have Turing machines $F_1, F_2, \ldots, F_n$ and G which will

compute these functions, we wish to design a machine H which will compute h. (All in embedded form as usual.)

Suppose we have code A for an m-tuple $(a_1, a_2, \ldots, a_m)$, we want the machine H to compute

$$X \ \square \ A \ \square \ Y \quad \rightarrow \quad X \ \square \ C \ \square \ Y$$

where C is the number code for $c = g(f_1(a_1, a_2, \ldots, a_m), f_2(a_1, a_2, \ldots, a_m), \ldots, f_n(a_1, a_2, \ldots, a_m))$. For convenience, let us write $B_1, B_2, \ldots, B_n$ for the codes for the numbers

$$f_1(a_1, a_2, \ldots, a_m)\,,\ f_2(a_1, a_2, \ldots, a_m)\,,\ \ldots\,,\ f_n(a_1, a_2, \ldots, a_m)\,.$$

The construction is not as difficult as one might expect. Starting with $\overset{\nabla}{X} \ \square \ A \ \square \ Y$,

$$\begin{aligned}
(k : l)\text{-duplicate } m\text{-tuple} \quad &\rightarrow \quad \overset{\nabla}{X} \ \square \ A \ \square \ A \ \square \ Y \\
(k : m + l)\text{-duplicate } m\text{-tuple} \quad &\rightarrow \quad \overset{\nabla}{X} \ \square \ A \ \square \ A \ \square \ A \ \square \ Y \\
&\vdots
\end{aligned}$$

and so on until we have $\overset{\nabla}{X} \ \square \ A \ \square \ A \ \square \ \ldots \ \square \ A \ \square \ Y$ with n copies of A. Next

$$\begin{aligned}
(k : (n-1)m + l)\text{-embedded form of } F_1 \quad &\rightarrow \quad \overset{\nabla}{X} \ \square \ B_1 \ \square \ A \ \square \ \ldots \ \square \ A \ \square \ Y \\
(k + 1 : (n-2)m + l)\text{-embedded form of } F_2 \quad &\rightarrow \quad \overset{\nabla}{X} \ \square \ B_1 \ \square \ B_2 \ \square \ \ldots \ \square \ A \ \square \ Y \\
&\vdots
\end{aligned}$$

and so on until all the As have been converted, ending with

$$\overset{\nabla}{X} \ \square \ B_1 \ \square \ B_2 \ \square \ \ldots \ \square \ B_n \ \square \ Y\,.$$

Finally

$$\begin{aligned}
(k : l)\text{-embedded form of } G \quad &\rightarrow \quad \overset{\nabla}{X} \ \square \ C \ \square \ Y \\
&\text{Finish}
\end{aligned}$$

C.17 Embedded minimalisation

Given a function $f : \mathbb{N}^{n+1} \rightarrow \mathbb{N}$, we define a function $g : \mathbb{N}^n \rightarrow \mathbb{N}$ by

$$g(a_1, a_2, \ldots, a_n) \ = \ \min_b \{\, f(b, a_1, a_2, \ldots, a_n) = 0 \,\}\,.$$

Assume that we are given a Turing machine F to compute f, we construct one, G say, to compute g (both in embedded form as usual).

$$\overset{\nabla}{X} \ \square \ A \ \square \ Y \quad \rightarrow \quad \overset{\nabla}{X} \ \square \ C \ \square \ Y\,,$$

where X is a k-tuple, Y an l-tuple, A the n-tuple code for $a_1, a_2, \ldots, a_n$ and C the number code for $g(a_1, a_2, \ldots, a_n)$.

$$\begin{aligned} \text{Step right across } k\text{-tuple} &\rightarrow X\ \square\ \overset{\triangledown}{A}\ \square\ Y \\ \text{Shift } (n+l)\text{-tuple one cell to right} &\rightarrow X\ \square\ \overset{\triangledown}{\square}\ A\ \square\ Y \end{aligned}$$

(Think of this as $X\ \square\ \overset{\triangledown}{B}\ \square\ A\ \square\ Y$ with B empty, i.e. code for 0. The true value of B will be developed by search.)

$$\begin{aligned} \text{Step left across } k\text{-tuple} &\rightarrow \overset{\triangledown}{X}\ \square\ B\ \square\ A\ \square\ Y \\ \text{(1) Duplicate } n+1\text{-tuple} &\rightarrow \overset{\triangledown}{X}\ \square\ B\ \square\ A\ \square\ B\ \square\ A\ \square\ Y \\ \text{Compute } F\text{, (k+1+n:l)-embedded} &\rightarrow \overset{\triangledown}{X}\ \square\ B\ \square\ A\ \square\ C\ \square\ Y \end{aligned}$$

(Here C is the number code for $f(b, a_1, a_2, \ldots, a_n)$.)

$$\begin{aligned} \text{Step right over } (k+1+n)\text{-tuple} &\rightarrow X\ \square\ B\ \square\ A\ \square\ \overset{\triangledown}{C}\ \square\ Y \\ \text{If looking at blank, go to (2), else } \ldots & \\ \text{Delete number} &\rightarrow X\ \square\ B\ \square\ A\ \square\ \overset{\triangledown}{Y} \\ \text{Step left across } (n+1)\text{-tuple} &\rightarrow X\ \square\ \overset{\triangledown}{B}\ \square\ A\ \square\ Y \\ \text{Shift } (n+1+l)\text{-tuple one cell to right} &\rightarrow X\ \square\ \overset{\triangledown}{\square}\ B\ \square\ A\ \square\ Y \\ \text{Write 1} &\rightarrow X\ \square\ \overset{\triangledown}{1}\ B\ \square\ A\ \square\ Y \\ \text{Step left across } k\text{-tuple} &\rightarrow \overset{\triangledown}{X}\ \square\ 1\ B\ \square\ A\ \square\ Y \\ \text{and go to (1)} & \end{aligned}$$

(2) If we get to here we have $X\ \square\ B\ \square\ A\ \square\ \overset{\triangledown}{\square}\ Y$ and B is code for the true value. Now tidy up:

$$\begin{aligned} \text{Step left over } (k+2+n)\text{-tuple} &\rightarrow \overset{\triangledown}{X}\ \square\ B\ \square\ A\ \square\ \square\ Y \\ \text{Delete } (n+1)\text{-tuple, (k+1:l) embedded} &\rightarrow \overset{\triangledown}{X}\ \square\ B\ \square\ Y \\ \text{Finish} & \end{aligned}$$

C.18 Theorem

Any partial recursive function is computable by a Turing machine whose alphabet has only two members.

Proof. The basic functions and operations defining partial recursive functions, as given in ▶ 8.A.2 **8**.A.2, have all been shown in this section to be computable by such a machine. ■

C.19 What next?

1 The most important result not proved here is that any function which is Turing computable (by a machine with any alphabet) is partial recursive. This will be left as a challenging exercise.

2 It is interesting, but inessential for our purposes, to prove things such as

- A Turing machine with a multi-dimensional tape is no more powerful: the functions that can be computed by such a machine are still just the partial recursive ones.
- A similar result for Turing machines with multiple tapes.

3 Traditionally, Turing machines have been used for much the same purposes as our toy ▶ Ch.**8** computing language of Chapter **8** was used in Chapters **8** and **9**. It should be obvious by now ▶ Ch.**9** that programming a Turing machine to perform any complicated operation is enormously tedious; this is why the toy computer language was used in these notes. It is also why, in the literature, when a Turing machine is being invoked to demonstrate the computability of some function, a good deal of "hand waving" usually occurs.

4 Probably the first really important result in the study of Turing machines was the construction of a *universal* Turing machine. This is a machine, U say, which, if supplied with a coded description of any other machine T on its tape followed by the code for an n-tuple will compute whatever T would have computed given that n-tuple as input.

This machine is used theoretically to establish the same sort of results that we have used ▶ **8**.E.2 the function $\varphi : \mathbb{N}^2 \to \mathbb{N}$ in **8**.E.2 for.

E. INDEX

Set theory: functions and constants

B

C

D

F

G

M

N

O

Q

R

S

T

U

V

W

X

Y

Z