CHAPTER ONE

Sentential Logic

§ 1.0 INFORMAL REMARKS ON FORMAL LANGUAGES

We are about to construct (in the next section) a language into which we can translate English sentences. Unlike natural languages (like English or Chinese), it will be a formal language, with precise formation rules. But before the precision begins, we will discuss here some of the features we want to incorporate into the language.

As a first example, the English sentence "Traces of potassium were observed" can be translated into the formal language as, say, the symbol **K**. Then for the closely related sentence "Traces of potassium were not observed," we can use **(¬K)**. Here ¬ is our negation symbol, read as "not." One might also think of translating "Traces of potassium were not observed" by some new symbol, e.g., **J**, but we will prefer instead to break such a sentence down into atomic parts as much as possible. For an unrelated sentence, "The sample contained chlorine," we choose, say, the symbol **C**. Then the following compound English sentences can be translated as the formulas shown at the right:

If traces of potassium were observed, then the sample did not contain chlorine.	**(K → (¬C))**
The sample contained chlorine, and traces of potassium were observed.	**(C ∧ K)**

In the second example we use our conjunction symbol ∧ to translate "and." The first example uses the more familiar arrow to translate "if . . . , then . . . ." In the following example the disjunction symbol ∨ is used to translate "or":

Either no traces of potassium were observed, or the sample did not contain chlorine.	**((¬K) ∨ (¬C))**
The sample neither contained chlorine nor were traces of potassium observed.	**(¬(C ∨ K))** or **((¬C) ∧ (¬K))**

In this last example we have given two alternative translations. The relationship between them will be discussed later.

One important aspect of the decompositions we will make of compound sentences is that whenever we are given the truth or falsity of the atomic parts, we can then immediately calculate the truth or falsity of the compound. Suppose, for example, that the chemist emerges from his laboratory and announces that he observed traces of potassium but that the sample contained no chlorine. We then know that the four above sentences are true, false, true, and false, respectively. In fact, we can construct in advance a table giving the four possible experimental results (Table I). We will return to the discussion of such tables in Section 1.3.

TABLE I

K	C	(¬(C ∨ K))	((¬C) ∧ (¬K))
F	*F*	*T*	*T*
F	*T*	*F*	*F*
T	*F*	*F*	*F*
T	*T*	*F*	*F*

Use of formal languages will allow us to escape from the imprecision and ambiguities of natural languages. But this is not done without cost; our formal languages will have a sharply limited degree of expressiveness.

In order to describe a formal language we will generally give three pieces of information:

1. We will specify the set of symbols (the alphabet). In the present case of sentential logic some of the symbols are

$$(\ ),\ \rightarrow,\ \neg,\ \mathbf{A}_1,\ \mathbf{A}_2,\ \ldots\ .$$

2. We will specify the rules for forming the "grammatically correct" finite sequences of symbols. (Such sequences will be called *well-formed formulas* or *wffs*.) For example, in the present case

$$(\mathbf{A}_1 \rightarrow (\neg \mathbf{A}_2))$$

will be a wff, whereas

$$)) \rightarrow \mathbf{A}_3$$

will not.

3. We will also indicate the allowable translations between English and the formal language. The symbols $\mathbf{A}_1$, $\mathbf{A}_2$, ... can be translations of declarative English sentences.

Only in this third part is any meaning assigned to the wffs. This process of assigning meaning guides and motivates all we do. But it will also be observed that it would theoretically be possible to carry out various manipulations with wffs in complete ignorance of any possible meaning. A person aware of only the first two pieces of information listed above could perform some of the things we will do, but it would make no sense to him.

Before proceeding, let us look briefly at another class of formal languages of widespread interest today. These are the languages used by (or at least in connection with) digital computers.

There are many of these languages. In one of them a typical wff is

01101011010100011111000100000111101 0.

In another a typical wff is

STEP#ADDIMAX,A.

(Here # is a symbol called a blank; it is brought into the alphabet so that a wff will be a string of symbols.) A well-known language called Fortran has wffs such as

DO#3#I=IMIN,IMAX.

In all cases there is a given way of translating the wffs into English, and (for a restricted class of English sentences) a way of translating from English into the formal language. But the computer is unaware of the English language. An unthinking automaton, the computer juggles symbols and follows its program slavishly. We could approach formal languages that way too, but it would not be much fun.

§ 1.1 THE LANGUAGE OF SENTENTIAL LOGIC

We assume we are given an infinite sequence of distinct objects which we will call symbols, and to which we now give names (Table II). We further assume that no one of these symbols is a finite sequence of other symbols.

TABLE II

Symbol	Verbose name	Remarks
(	left parenthesis	punctuation
)	right parenthesis	punctuation
$\neg$	negation symbol	English: not
$\wedge$	conjunction symbol	English: and
$\vee$	disjunction symbol	English: or (inclusive)
$\rightarrow$	conditional symbol	English: if__, then__
$\leftrightarrow$	biconditional symbol	English: if and only if
$\mathbf{A}_1$	first sentence symbol	
$\mathbf{A}_2$	second sentence symbol	
...		
$\mathbf{A}_n$	nth sentence symbol	
...		

Several remarks are now in order:

1. The five symbols

$$\neg,\ \wedge,\ \vee,\ \rightarrow,\ \leftrightarrow$$

are called *sentential connective* symbols. Their use is suggested by the English translation given above. The sentential connective symbols, together with the parentheses, are the *logical symbols*. In translating to and from English, they always play the same role. The sentence symbols are the *parameters* (or *nonlogical symbols*). Their translation is not fixed; instead they will be open to a variety of interpretations, as we shall illustrate shortly.

1 ταχυδακτυλουργῶ

2. We have included only countably many sentence symbols. Most of the things we say in this chapter would apply equally well if we had allowed an arbitrary set of sentence symbols. (The exceptions are primarily in Section 1.7.)

3. Some logicians prefer to call $\mathbf{A}_n$ the nth *proposition* symbol (and to speak of propositional logic instead of sentential logic). This stems from wanting the word "sentence" to refer to a particular utterance, and wanting a proposition to be that which a sentence asserts.

4. We call these objects "symbols," but we remain neutral as to what the exact ontological status of symbols might be. In the leftmost column of our list of symbols, names of the symbols are printed. For example, $\mathbf{A}_{243}$ is one symbol, namely the two hundred forty-third sentence symbol. (On the other hand, "$\mathbf{A}_{243}$" is a name of that symbol. The conditional symbol may or may not have the geometric property of being shaped like an arrow, although its name "$\rightarrow$" does.) The symbols themselves can be sets, numbers, marbles, or objects from a universe of linguistic objects. In the last case, it is conceivable that they are actually the same things as the names we use for them. Another possibility, which will be exploited in the next chapter, is that the sentence symbols are themselves formulas in another language.

5. We have assumed that no symbol is a finite sequence of other symbols. We mean by this that not only are the symbols listed distinct (e.g., $\mathbf{A}_3 \neq \leftrightarrow$), but no one of them is a finite sequence of two or more symbols. For example, we demand that $\mathbf{A}_3 \neq \langle \neg, \mathbf{A}_4, (\rangle$. (It is a fact of set theory that $\mathbf{A}_3 \neq \langle \neg, \mathbf{A}_3, (\rangle$.) The purpose of this assumption is to assure that finite sequences of symbols be uniquely decomposable. If

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

and each a_i and each b_j is a symbol, then $m = n$ and $a_i = b_i$. (See Chapter 0, Lemma 0A, and subsequent remarks.)

An *expression* is a finite sequence of symbols. We can specify an expression by concatenating the names of the symbols; thus $(\neg\mathbf{A}_1)$ is the sequence $\langle (, \neg, \mathbf{A}_1,) \rangle$. This notation is extended: If α and β are sequences of symbols, then $\alpha\beta$ is the sequence consisting first of the symbols of the sequence α followed by the symbols of the sequence β.

For example, if α and β are the expressions given by the equations

$$\alpha = (\neg\mathbf{A}_1),$$
$$\beta = \mathbf{A}_2,$$

then $(\alpha \rightarrow \beta)$ is the expression

$$((\neg\mathbf{A}_1) \rightarrow \mathbf{A}_2).$$

We should now look at a few examples of possible translations of English sentences into expressions of the formal language. Let $\mathbf{A}, \mathbf{B}, \ldots, \mathbf{Z}$ be the first twenty-six sentence symbols. (For example, $\mathbf{E} = \mathbf{A}_5$.)

1. English: The suspect must be released from custody. Translation: **R**.
English: The evidence obtained is admissible. Translation: **E**.
English: The evidence obtained is inadmissible. Translation: $(\neg\mathbf{E})$.
English: The evidence obtained is admissible, and the suspect need not be released from custody. Translation: $(\mathbf{E} \wedge (\neg\mathbf{R}))$.
English: Either the evidence obtained is admissible, or the suspect must be released from custody (or possibly both). Translation: $(\mathbf{E} \vee \mathbf{R})$.
English: Either the evidence obtained is admissible, or the suspect must be released from custody, but not both. Translation: $((\mathbf{E} \vee \mathbf{R}) \wedge (\neg(\mathbf{E} \wedge \mathbf{R})))$. We intend always to use the symbol $\vee$ to translate the word "or" in its inclusive meaning of "and/or."
English: The evidence obtained is inadmissible, but the suspect need not be released from custody. Translation: $((\neg\mathbf{E}) \wedge (\neg\mathbf{R}))$. On the other hand, the expression $((\neg\mathbf{E}) \vee (\neg\mathbf{R}))$ translates the English: Either the evidence obtained is inadmissible or the suspect need not be released from custody.

2. English: If wishes are horses, then beggars will ride. Translation: $(\mathbf{W} \rightarrow \mathbf{B})$.
English: Beggars will ride if and only if wishes are horses. Translation: $(\mathbf{B} \leftrightarrow \mathbf{W})$.

3. English: This commodity constitutes wealth if and only if it is transferable, limited in supply, and either productive of pleasure or preventive of pain. Translation: $(\mathbf{W} \leftrightarrow (\mathbf{T} \wedge (\mathbf{L} \wedge (\mathbf{P} \vee \mathbf{Q}))))$. Here **W** is the translation of "This commodity constitutes wealth." Of course in the preceding example we used **W** to translate a different sentence. We are not tied to any one translation.

One note of caution: Do not confuse a sentence in the English language (Roses are red) with a translation of that sentence in the formal language (e.g., **R**). These are different. The English sentence is presumably either true or false. But the formal expression is just a sequence of symbols. It may indeed be interpreted in one context as a true (or false) English sentence, but it can have other interpretations in other contexts.

Now some expressions cannot be obtained as translations of any English sentences but are mere nonsense, like

$$\rightarrow ((A_3.$$

We want to define the well-formed formulas (wffs) to be the "grammatically correct" expressions; the nonsensical expressions must be excluded. The definition will have the following consequences:

(a) Every sentence symbol is a wff.

(b) If α and β are wffs, then so are $(\neg\alpha)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$.

(c) No expression is a wff unless it is compelled to be one by (a) and (b).

There are two equivalent ways of making this third property (about compulsion) precise. The first way defines the set of wffs "from the top down." Let us say that a set S of expressions is *inductive* iff it has the properties

(a_S) Every sentence symbol is in S.

(b_S) If expressions α and β are in S, then so also are $(\neg\alpha)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$.

An expression is a *well-formed formula* (*wff*) iff it is a member of every inductive set. Then it is not hard to see that (a) and (b) are satisfied. As for (c), we can say that the set of all wffs is as small as it can be, in the sense that it is a subset of any other inductive set.

The second (and equivalent) definition works "from the bottom up." An expression is a wff if and only if it can be built up from the sentence symbols by applying some finite number of times the *formula-building operations* (on expressions) defined by the equations

$$\mathscr{E}_\neg(\alpha) = (\neg\alpha),$$

$$\mathscr{E}_\wedge(\alpha, \beta) = (\alpha \wedge \beta),$$

$$\mathscr{E}_\vee(\alpha, \beta) = (\alpha \vee \beta),$$

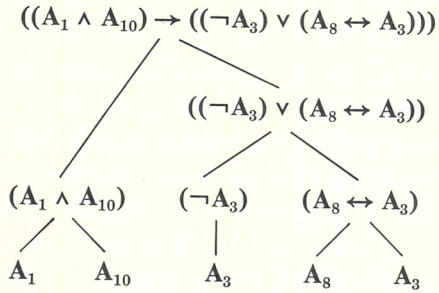
$$\mathscr{E}_\rightarrow(\alpha, \beta) = (\alpha \rightarrow \beta),$$

$$\mathscr{E}_\leftrightarrow(\alpha, \beta) = (\alpha \leftrightarrow \beta).$$

For example,

$$((A_1 \wedge A_{10}) \rightarrow ((\neg A_3) \vee (A_8 \leftrightarrow A_3)))$$

is a wff, as can be seen by contemplating its ancestral tree:



The tree illustrates the construction of the expression from four sentence symbols by five applications of formula-building operations.

These two definitions of wffs are discussed in a more general setting in Section 1.2. Here we only note that any inductive set of wffs is actually the set of all wffs; this fact will be called the *induction principle*. This principle will receive much use in the coming pages. In the following example we use it to show that certain expressions are *not* wffs.

EXAMPLE Any expression with more left parentheses than right parentheses is not a wff.

Proof The idea is that we start with sentence symbols (having zero left parentheses and zero right parentheses), and then apply formula-building operations which add parentheses only in matched pairs. We can rephrase this argument as follows: The set of "balanced" wffs (having equal numbers of left and right parentheses) contains all sentence symbols and is closed under the formula-building operations. Thus the set of balanced wffs is inductive; the induction principle then assures us that all wffs are balanced. ■

EXERCISES

1. Give three sentences in English together with translations into our formal language. The sentences should be chosen so as to have an interesting structure, and the translations should each contain fifteen or more symbols.

2. Show that there are no wffs of length 2, 3, or 6, but that any other length is possible.

3. Let α be a wff; let c be the number of places at which binary connective symbols ($\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$) occur in α; let s be the number of places at which sentence symbols occur in α. Show that $s = c + 1$.

§ 1.2 INDUCTION AND RECURSION

Induction

There is one special type of construction which occurs frequently both in logic and in other branches of mathematics. We may want to construct a certain subset of a set U by starting with some initial elements of U, and applying certain operations to them over and over again. The set we seek will be the smallest set containing the initial elements and closed under the operations. Its members will be those elements of U which can be built up from the initial elements by applying the operations a finite number of times.

In the special case of immediate interest to us, U is the set of expressions, the initial elements are the sentence symbols, and the operations are $\mathscr{E}_\neg$, $\mathscr{E}_\wedge$, etc. The set to be constructed is the set of wffs. But we will encounter other special cases later, and it will be helpful to view the situation abstractly here.

To simplify our discussion, we will consider an initial set $B \subseteq U$ and a class $\mathscr{F}$ of functions containing just two members f and g, where

$$f : U \times U \to U \quad \text{and} \quad g : U \to U.$$

Thus f is a binary operation on U and g is a unary operation. (Actually $\mathscr{F}$ need not be finite; it will be seen that our simplified discussion here is, in fact, applicable to a more general situation. $\mathscr{F}$ can be any set of relations on U, and in Chapter 2 this greater generality will be utilized. But the case discussed here is easier to visualize and is general enough to illustrate the ideas. For a less restricted version, see Exercise 3.)

If B contains points a and b, then the set C we wish to construct will contain, for example,

$$b, f(b, b), g(a), f(g(a), f(b, b)), g(f(g(a), f(b, b))).$$

Of course these might not all be distinct. The idea is that we are given certain bricks to work with, and certain types of mortar, and we want C to contain just the things we are able to build.

In defining C more formally, we have our choice of two definitions. We can define it "from the top down" as follows: Say that a subset S of U is *closed* under f and g iff whenever elements x and y belong to S, then so do $f(x, y)$ and $g(x)$. Say that S is *inductive* iff $B \subseteq S$ and S is closed under f and g. Let C^* be the intersection of all the inductive subsets of U; thus

$x \in C^*$ iff x belongs to every inductive subset of U. It is not hard to see (and the reader should check) that C^* is itself inductive. Furthermore, C^* is the smallest such set, being included in all the other inductive sets.

The second (and equivalent) definition works "from the bottom up." We want C_* to contain the things which can be reached from B by applying f and g a finite number of times. Temporarily define a *construction sequence* to be a finite sequence $\langle x_0, \ldots, x_n \rangle$ of elements of U such that for each $i \leq n$ we have at least one of

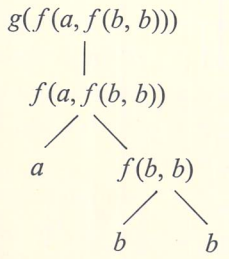
$$\begin{aligned} &x_i \in B, \\ &x_i = f(x_j, x_k) \qquad \text{for some} \quad j < i,\ k < i, \\ &x_i = g(x_j) \qquad \text{for some} \quad j < i. \end{aligned}$$

Then let C_* be the set of all points x such that some construction sequence ends with x.

Let C_n be the set of points x such that some construction sequence of length n ends with x. Then $C_1 = B$,

$$C_1 \subseteq C_2 \subseteq C_3 \subseteq \cdots,$$

and $C_* = \bigcup_n C_n$. For example, $g(f(a, f(b, b)))$ is in C_5 and hence in C_*, as can be seen by contemplating the tree shown:



We obtain a construction sequence for $g(f(a, f(b, b)))$ by squashing this tree into a linear ordering.

EXAMPLES 1. The natural numbers. Let U be the set of all real numbers, and let $B = \{0\}$. Take one operation S, where $S(x) = x + 1$. Then

$$C_* = \{0, 1, 2, \ldots\}.$$

The set C_* of natural numbers contains exactly those numbers obtainable from 0 by applying the successor operation repeatedly.

2. The integers. Let U be the set of all real numbers; let $B = \{0\}$. This time take two operations, the successor operation S and the predecessor operation P:

$$S(x) = x + 1 \quad \text{and} \quad P(x) = x - 1.$$

Now C_* contains all the integers,

$$C_* = \{\ldots, -2, -1, 0, 1, 2, \ldots\}.$$

Notice that there is more than one way of obtaining 2 as a member of C_*. For 2 is $S(S(0))$, but it is also $S(P(S(S(0))))$.

3. The algebraic functions. Let U contain all functions whose domain and range are each sets of real numbers. Let B contain the identity function and all constant functions. Let $\mathscr{F}$ contain the operations (on functions) of addition, multiplication, division, and root extraction. Then C_* is the class of algebraic functions.

4. The well-formed formulas. Let U be the set of all expressions and let B be the set of sentence symbols. Let $\mathscr{F}$ contain the five formula-building operations on expressions: $\mathscr{E}_\neg$, $\mathscr{E}_\wedge$, $\mathscr{E}_\vee$, $\mathscr{E}_\to$, and $\mathscr{E}_\leftrightarrow$. Then C_* is the set of all wffs.

At this point we should verify that our two definitions are actually equivalent, i.e., that $C^* = C_*$.

To show that $C^* \subseteq C_*$ we need only check that C_* is inductive, i.e., that $B \subseteq C_*$ and C_* is closed under the functions. Clearly $B = C_1 \subseteq C_*$. If x and y are in C_*, then we can concatenate their construction sequences and append $f(x, y)$ to obtain a construction sequence placing $f(x, y)$ in C_*. Similarly, C_* is closed under g.

Finally, to show that $C_* \subseteq C^*$ we consider a point in C_* and a construction sequence $\langle x_0, \ldots, x_n\rangle$ for it. By ordinary induction on i, we can see that $x_i \in C^*$, $i \leq n$. First $x_0 \in B \subseteq C^*$. For the inductive step we use the fact that C^* is closed under the functions. Thus we conclude that

$$\bigcup_n C_n = C_* = C^* = \bigcap\{S : S \text{ is inductive}\}.$$

(A parenthetical remark: Suppose our present study is embedded in axiomatic set theory, where the natural numbers are usually defined from the top down. Then our definition of C_* (employing finiteness and hence natural numbers) is not really different from our definition of C^*. But we are not working within axiomatic set theory; we are working within

1 γράφω στὸ τέλος

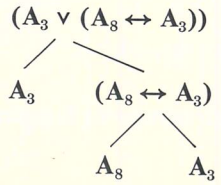
intuitive mathematics. And the notion of natural number seems to be a solid, well-understood intuitive concept.)

Since $C^* = C_*$, we call the set simply C and refer to it as the set *generated from B by* the functions in $\mathscr{F}$. We will often want to prove theorems by using the following:

Induction Principle Assume that C is the set generated from B by the functions in $\mathscr{F}$. If S is a subset of C which includes B and is closed under the functions of $\mathscr{F}$, then $S = C$.

Proof S is inductive, so $C = C^* \subseteq S$. We are given the other inclusion. ■

The special case now of interest to us is, of course, Example 4. Here C is the class of wffs generated from the set of sentence symbols by the formula-building operations. This special case has interesting features of its own. Both α and β are proper segments of $\mathscr{E}_\wedge(\alpha, \beta)$, i.e., of $(\alpha \wedge \beta)$. More generally, if we look at the family tree of a wff, we see that each constituent is a proper segment of the end product.



Suppose, for example, that we temporarily call an expression *special* if the only sentence symbols in it are among $\{\mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_5\}$ and the only connective symbols in it are among $\{\neg, \to\}$. Then no special wff requires $\mathbf{A}_9$ or $\mathscr{E}_\wedge$ for its construction. In fact, every special wff belongs to the set C_s generated from $\{\mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_5\}$ by $\mathscr{E}_\neg$ and $\mathscr{E}_\to$. (For we can use the induction principle to show that every wff either belongs to C_s or is not special.)

Recursion[1]

We return now to the more abstract case. There is a set U (such as the set of all expressions), a subset B of U (such as the set of sentence symbols),

[1] The reader not already familiar with recursion is advised to postpone reading this subsection until after reading Section 1.3, where a specific application is encountered.

and two functions f and g, where

$$f : U \times U \to U \quad \text{and} \quad g : U \to U.$$

C is the set generated from B by f and g.

The problem we now want to consider is that of defining a function on C recursively. That is, we suppose we are given

1. Rules for computing $\bar{h}(x)$ for $x \in B$.
2a. Rules for computing $\bar{h}(f(x, y))$, making use of $\bar{h}(x)$ and $\bar{h}(y)$.
2b. Rules for computing $\bar{h}(g(x))$, making use of $\bar{h}(x)$.

(For example, this is the situation discussed in Section 1.3, where $\bar{h}$ is the extension of a truth assignment for B.) It is not hard to see that there can be at most one function $\bar{h}$ on C meeting all the given requirements.

But it is possible that no such $\bar{h}$ exists; the rules may be contradictory. For example, let

$$U = \text{the set of real numbers},$$
$$B = \{0\},$$
$$f(x, y) = x \cdot y,$$
$$g(x) = x + 1.$$

Then C is the set of natural numbers. Suppose we impose the following requirements on $\bar{h}$:

1. $\bar{h}(0) = 0$.
2a. $\bar{h}(f(x, y)) = f(\bar{h}(x), \bar{h}(y))$.
2b. $\bar{h}(g(x)) = \bar{h}(x) + 2$.

Then no such function $\bar{h}$ can exist. (Try computing $\bar{h}(1)$, noting that we have both $1 = g(0)$ and $1 = f(g(0), g(0))$.

A similar situation is encountered in algebra.[1] Suppose that you have a group G which is generated from B by the group multiplication and inverse operation. Then an arbitrary map of B into a group H is not necessarily extendible to a homomorphism of the entire group G into H. But if G happens to be a free group with set B of independent generators, then any such map is extendible to a homomorphism of the entire group.

[1] It is hoped that examples such as this will be useful to the reader with some algebraic experience. The other readers will be glad to know that these examples are merely illustrative and not essential to our development of the subject.

Say that C is *freely* generated from B by f and g iff in addition to the requirements for being generated we have

1. f_C and g_C are one-to-one, and
2. The range of f_C, the range of g_C, and the set B are pairwise disjoint.

(Here f_C and g_C are the restrictions of f and g to C.)

Recursion Theorem Assume that the subset C of U is freely generated from B by f and g, where

$$f : U \times U \to U,$$
$$g : U \to U.$$

Further assume that V is a set and F, G, and h functions such that

$$h : B \to V,$$
$$F : V \times V \to V,$$
$$G : V \to V.$$

Then there is a unique function

$$\bar{h} : C \to V$$

such that

(i) For x in B, $\bar{h}(x) = h(x)$.
(ii) For x, y in C,

$$\bar{h}(f(x, y)) = F(\bar{h}(x), \bar{h}(y)),$$
$$\bar{h}(g(x)) = G(\bar{h}(x)).$$

Viewed algebraically, the conclusion of this theorem says that any map h of B into V can be extended to a homomorphism $\bar{h}$ from C (with operations f and g) into V (with operations F and G).

If the content of the recursion theorem is not immediately apparent, try looking at it chromatically. You want to have a function $\bar{h}$ which paints each member of C some color. You have before you

1. h, telling you how to color the initial elements in B;
2. F, which tells you how to combine the color of x and y to obtain the color of $f(x, y)$ (i.e., it gives $\bar{h}(f(x, y))$ in terms of $\bar{h}(x)$ and $\bar{h}(y)$);
3. G, which similarly tells you how to convert the color of x into the color of $g(x)$.

1 μεταγράφω

The danger is that of a conflict in which, for example, F is saying "green" but G is saying "red" for the same point (unlucky enough to be equal both to $f(x, y)$ and $g(z)$ for some x, y, z). But if C is freely generated, then this danger is avoided.

EXAMPLES Consider again the examples of the preceding subsection.

1. $B = \{0\}$ with one operation, the successor operation S. Then C is the set $\mathbb{N}$ of natural numbers. Since the successor operation is one-to-one, C is freely generated from $\{0\}$ by S. Therefore, by the recursion theorem, for any set V, any $a \in V$, and any $F : V \to V$ there is a unique $\bar{h} : \mathbb{N} \to V$ such that $\bar{h}(0) = a$ and $\bar{h}(S(x)) = F(\bar{h}(x))$ for each $x \in \mathbb{N}$. For example, there is a unique $\bar{h} : \mathbb{N} \to \mathbb{N}$ such that $\bar{h}(0) = 0$ and $\bar{h}(S(x)) = 1 - \bar{h}(x)$. This function has the value 0 at even numbers and the value 1 at odd numbers.

2. The integers are generated from $\{0\}$ by the successor and predecessor operations but not freely generated.

3. Freeness fails also for the generation of the algebraic functions in the manner described.

4. The wffs are freely generated from the sentence symbols by the five formula-building operations. The purpose of Section 1.4 is to prove this fact. It follows, for example, that there is a unique function $\bar{h}$ defined on the set of wffs such that

$$\bar{h}(A) = 1 \text{ for a sentence symbol } A,$$
$$\bar{h}((\neg\alpha)) = 3 + \bar{h}(\alpha),$$
$$\bar{h}((\alpha \wedge \beta)) = 3 + \bar{h}(\alpha) + \bar{h}(\beta),$$

and similarly for $\vee$, $\to$, and $\leftrightarrow$. This function gives the length of each wff.

Proof of the recursion theorem The idea is to let $\bar{h}$ be the union of many approximating functions. Temporarily call a function v (which maps part of C into V) *acceptable* if it meets the conditions imposed on $\bar{h}$ by (i) and (ii). More precisely, v is acceptable iff the domain of v is a subset of C, the range a subset of V, and

(i′) If x belongs to B and to the domain of v, then $v(x) = h(x)$.

(ii′) If $f(x, y)$ belongs to the domain of v, then so do x and y, and $v(f(x, y)) = F(v(x), v(y))$. If $g(x)$ belongs to the domain of v, then so does x, and $v(g(x)) = G(v(x))$.

Let K be the collection of all acceptable functions, and let $\bar{h}$ be the union of K. Thus

$$\langle x, y\rangle \in \bar{h} \quad \text{iff } v(x) = y \text{ for some } v \in K.$$

We claim that $\bar{h}$ meets our requirements. We will outline the procedure for checking this, leaving many details to the reader. (We feel that a detailed understanding of this set-theoretic proof, while nice, is not essential here. But some understanding of its outline should make the theorem itself more comprehensible.)

1. We claim that $\bar{h}$ is a function. Let

$$S = \{x \in C\text{: For at most one } y, \langle x, y\rangle \in \bar{h}\}.$$

It is easy to verify that S is inductive, by using (i′) and (ii′). Hence $S = C$ and $\bar{h}$ is a function.

2. We claim that $\bar{h} \in K$; i.e., that $\bar{h}$ is an acceptable function. This follows fairly easily from the definition of $\bar{h}$ and the fact that it is a function.

3. We claim that $\bar{h}$ is defined throughout C. It suffices to show that the domain of $\bar{h}$ is inductive. It is here that the assumption of freeness is used. For example, one case is the following: Suppose that x is in the domain of $\bar{h}$. Then $\bar{h} \cup \langle g(x), G(\bar{h}(x))\rangle$ is acceptable. (The freeness is required in showing that it is a function.) Consequently, $g(x)$ is in the domain of $\bar{h}$.

4. We claim that $\bar{h}$ is unique. For given two such functions, let S be the set on which they agree. Then S is inductive, and so equals C. ■

It is interesting to note that there is an alternative way of describing the proof of the recursion theorem, by presenting the desired function $\bar{h}$ as the set (of pairs) generated from a set by some operations. For let

$$\hat{U} = U \times V,$$
$$\hat{B} = h, \quad \text{a subset of } \hat{U},$$
$$\hat{F}(\langle x, u\rangle, \langle y, v\rangle) = \langle f(x, y), F(u, v)\rangle,$$
$$\hat{G}(\langle x, u\rangle) = \langle g(x), G(u)\rangle.$$

Thus $\hat{F}$ is the binary operation on $\hat{U}$ obtained as the product of the operations f and F. Now let $\bar{h}$ be the subset of $\hat{U}$ generated from $\hat{B}$ by $\hat{F}$ and $\hat{G}$. Then it can be checked that $\bar{h}$ has the desired properties. The freeness must be used in showing that $\bar{h}$ is a function.

One final comment on induction and recursion: The induction principle we have stated is not the only one possible. It is entirely possible to give

proofs by induction (and definitions by recursion) on the length of expressions, the number of places at which connective symbols occur, etc. Such methods are inherently less basic but may be necessary in some situations.

EXERCISES

1. Suppose that C is generated from a set $B = \{a, b\}$ by the binary operation f and unary operation g. List all the members of C_2. How many members might C_3 have? C_4?

2. Obviously $(\mathbf{A}_3 \rightarrow \wedge\, \mathbf{A}_1)$ is not a wff. But prove that it is not a wff.

3. We can generalize the discussion in this section by requiring of $\mathscr{F}$ only that it be a class of relations on U. C_* is defined as before, except that $\langle x_0, x_1, \ldots, x_n\rangle$ is now a construction sequence provided that for each $i \leq n$ we have either $x_i \in B$ or $\langle x_{j_1}, \ldots, x_{j_k}, x_i\rangle \in R$ for some $R \in \mathscr{F}$ and some $j_1, \ldots, j_k$ all less than i. Give the correct definition of C^* and show that $C^* = C_*$.

§ 1.3 TRUTH ASSIGNMENTS

We want to define what it means for one wff of our language to follow logically from other wffs. For example, $\mathbf{A}_1$ should follow from $(\mathbf{A}_1 \wedge \mathbf{A}_2)$. For no matter how the parameters $\mathbf{A}_1$ and $\mathbf{A}_2$ are translated back into English, if the translation of $(\mathbf{A}_1 \wedge \mathbf{A}_2)$ is true, then the translation of $\mathbf{A}_1$ must be true. But the notion of all possible translations back into English is unworkably vague. Luckily the spirit of this notion can be expressed in a simple and precise way.

Fix once and for all a set $\{T, F\}$ of *truth values* consisting of two distinct points:

T, called *truth*,
F, called *falsity*.

(It makes no difference what these points themselves are; they might as well be the numbers 1 and 0.) Then a *truth assignment* v for a set $\mathscr{S}$ of sentence symbols is a function

$$v : \mathscr{S} \rightarrow \{T, F\}$$

assigning either T or F to each symbol in $\mathscr{S}$. These truth assignments will be used in place of the translations into English mentioned in the preceding paragraph.

(At this point we have committed ourselves to *two-valued* logic. It is also possible to study three-valued logic, in which case one has a set of three possible truth values. And then, of course, it is a small additional step to allow 512 or $\aleph_0$ truth values; or to take as the set of truth values the unit interval [0, 1] or some other convenient space. A particularly interesting case is that for which the truth values form a complete Boolean algebra. But it is two-valued logic that has always had the greatest significance, and we will be content to confine ourselves to this case.)

Let $\bar{\mathscr{S}}$ be the set of wffs generated from $\mathscr{S}$ by the five formula-building operations. ($\bar{\mathscr{S}}$ can also be characterized as the set of wffs whose sentence symbols are all in $\mathscr{S}$; see the remarks at the end of the subsection on induction in Section 1.2.) We want an extension $\bar{v}$ of v,

$$\bar{v} : \bar{\mathscr{S}} \rightarrow \{T, F\},$$

which assigns the correct truth value to each wff in $\bar{\mathscr{S}}$. It should meet the following conditions:

0. For any $A \in \mathscr{S}$, $\bar{v}(A) = v(A)$. (Thus $\bar{v}$ is an extension of v.)

For any α, β in $\bar{\mathscr{S}}$:

1. $\bar{v}((\neg\alpha)) = \begin{cases} T \text{ if } \bar{v}(\alpha) = F, \\ F \text{ otherwise.} \end{cases}$

2. $\bar{v}((\alpha \wedge \beta)) = \begin{cases} T \text{ if } \bar{v}(\alpha) = T \text{ and } \bar{v}(\beta) = T, \\ F \text{ otherwise.} \end{cases}$

3. $\bar{v}((\alpha \vee \beta)) = \begin{cases} T \text{ if } \bar{v}(\alpha) = T \text{ or } \bar{v}(\beta) = T \text{ (or both)}, \\ F \text{ otherwise.} \end{cases}$

4. $\bar{v}((\alpha \rightarrow \beta)) = \begin{cases} F \text{ if } \bar{v}(\alpha) = T \text{ and } \bar{v}(\beta) = F, \\ T \text{ otherwise.} \end{cases}$

5. $\bar{v}((\alpha \leftrightarrow \beta)) = \begin{cases} T \text{ if } \bar{v}(\alpha) = \bar{v}(\beta), \\ F \text{ otherwise.} \end{cases}$

(Conditions 1–5 are given in tabular form in Table III. It is at this point that the intended meaning of, for example, the conjunction symbol enters into our formal proceedings. Note especially the intended meaning of $\rightarrow$. Whenever α is assigned F, then $(\alpha \rightarrow \beta)$ is considered "vacuously true" and is assigned the value T. For this and the other connectives, it is certainly possible to question how accurately we have reflected the common meaning in everyday speech of "if . . . , then," "or," etc. But our ultimate concern lies more with mathematical statements than with the subtle nuances of everyday speech.)