SYMBOLIC LOGIC
AND
INTELLIGENT MACHINES

EDMUND C. BERKELEY
Berkeley Enterprises, Inc.
Newtonville, Massachusetts

Reinhold Publishing Corporation
New York
Chapman & Hall Ltd., London

Second Printing 1961

Library of Congress Catalog Card Number: 59-9903

Composition by The Science Press, Inc.

Printed in the United States of America
THE GUINN CO., INC.
New York 14, N. Y.

PREFACE

Intelligence, according to the dictionary, is the ability to learn or understand from experience, the ability to acquire and retain knowledge, and the ability to respond quickly and appropriately to new situations, resulting in success in using these abilities to perform tasks. These abilities are now possessed, at least partially, by some machines.

Of these machines the most notable perhaps are the great automatic computers. Some of them in fact can perform over 10,000 calculating operations per second, on numbers of more than 10 decimal digits. But there are many other kinds of automatic machines besides, that also behave differently and appropriately in response to various signals. It is reasonable to call all such machines *intelligent machines.*

The operation of intelligent machines depends on the on-ness or off-ness of signals and circuits—the pattern of interaction of yeses and noes. As a result, the science of dealing with patterns of interaction of yeses and noes has taken on fresh and considerable importance. This science is essentially *symbolic logic* rather than mathematics, because the emphasis is not on numerical relations but on non-numerical relations. For example, the statement: "If A is the father of B, and B is the father of C, then A is the grandfather of C," displays a non-numerical relation; and so does the statement: "If switch A is on and switch B is off, then the combination of A in series with B is off, but the combination of A in parallel with B is on". Symbolic logic in fact is being used more and more: in the handling of rules, clauses, and contracts; in designing circuits for computing machinery, telephone systems, and control devices; in programming automatic computers; and in describing and designing many types of circuits and mechanisms.

The main purpose of this book is to make clear the nature of symbolic logic and the properties of intelligent machines, and to show how these two subjects are related—to the advantage of both. It is also a purpose of this book to explain the application of symbolic logic in the design of machines that behave intelligently.

This book contains two kinds of material—material which should be of interest and use to most readers for most purposes, such as some of the explanation of Boolean algebra and the summary of the rules for calculating with it; and material which is difficult or supplementary—lists, tables, intricate diagrams, discussion of ideas without padding between them, etc., and which is included in this book for some readers for some purposes. Many readers will find these sections of the book unnecessary for their purposes and can omit them. This book is not intended to be read consecutively from start to finish without skipping.

The origin of this book goes back to a talk—I remember it as if it were yesterday—that I had in 1927 with a mathematics professor of mine, Dr. George D. Birkhoff, when I was a sophomore at Harvard. I remarked to him that I wished there were an algebra of language—an algebra by means of which one could calculate the answer to an argument instead of going to sea in an ocean of words. He said to me that there was, and told me of a book called "The Laws of Thought" by George Boole, published in 1854. I quickly sought the book in the library, and although it confused me, since I did not then know about later improvements in Boolean algebra, nevertheless, the fact that here was an algebra dealing with things like classes and statements, and operations like AND, OR, NOT, opened wide for me a most exhilarating vista—the possibility that all the language of thought could really become calculable like mathematics. This possibility has become more real than ever before because of the calculating powers of automatic computers. So I hope that the contents of this book may likewise stir the imagination and spirits of many readers.

It is too much to hope that every statement in this book is correct. I shall be very grateful for all additions, corrections, comments, and suggestions that any reader may be kind enough to send me.

February 15, 1959
Newtonville 60, Mass.

Edmund C. Berkeley

CONTENTS

Chapter 1

SYMBOLIC LOGIC: AN INTRODUCTION

1. What is Symbolic Logic About?

Whenever we approach a new subject, our first problem is finding out what the new subject is about, what it deals with. And to our surprise, we often find we have already had a good deal of experience with its content, even if we are strangers to its special vocabulary. We are like Molière's rich man in "Le Bourgeois Gentilhomme", who discovered that all his life he had been speaking "prose".

You and I and everybody we know have all our lives been dealing with much of the content of symbolic logic. We are not strangers to it: it is the underlying fabric of much of our thinking. We deftly and quite unconsciously adjust to many of its fine points. But nearly all of us are completely unaware of the modern science which deals with these ideas.

The content of symbolic logic consists of many of the commonest ideas expressed in the commonest words and phrases of language. A few of these words and phrases are shown in Table 1-1.

TABLE 1-1

the	there is	other	for	it
of	same	than	by	thing
some	different	a, an	with	kind
yes	is a	another	from	sort
no	as	to	has	which

All these words are commonly used by six-year olds. Expressed in more advanced words, words which would probably be understood by a senior in college, the content of symbolic logic includes the ideas expressed in the following words and phrases:

statements, sentences, propositions
truth, falsity, assertion, denial
reasoning, implications, theorems, proofs
individuals, elements, things

properties, relations
classes, groups, collections, types
choosing, selection, arrangement, comparison, matching, correspondence, merging, collating, sorting

But the words of ordinary language are often neither exact nor clear. We have much trouble saying just what we mean whenever we use a word all by itself, an isolated word, a word without a context surrounding it. For that word has many different meanings, and we are never sure, without more indications than the word alone, which meaning to give to it. Words suffer from being ambiguous. Take for example the word "yes": it is ambiguous in ordinary language. Some of the meanings of the word "yes" as it occurs in conversation are these:

"yes" no. 1: The statement referred to is true.
"yes" no. 2: I don't want to disagree with you in public.
"yes" no. 3: probably
"yes" no. 4: maybe, perhaps
"yes" no. 5: I did not hear what you said, but I want to be polite.

Because of the ambiguity of the common words of ordinary language, we not only find it hard to know exactly what such a word means when someone else uses it, but also not to mislead ourselves when we ourselves use it.

To achieve precise meaning, the scientists working in the field of symbolic logic and reasoning—the symbolic logicians—have invented symbols and given them precise meanings.

Each symbol invented selects, codifies, signals, one particular meaning out of many meanings of such words. When that symbol is used again, precisely that meaning is meant. And that meaning is fixed, stabilized, and sharpened by careful exact (calculating) relation to other exact symbols. For example, in symbolic logic "the" is coupled with the assertion "there exists one and only one". For example, when you say "the Statue of Liberty", you also imply the assertion that "there is one and only one Statue of Liberty".

As a result of the precise meanings of the symbols, and the calculating relationships among them, we gain an enormous clarity—such a wonderful clarity that many ideas we were unable to think of or realize or express in ordinary language become expressible and open to study and understanding. In this way, the foundations of mathematics have become far better understood than ever before.

A good symbolism that precisely suits a field of thought becomes an indispensable tool for working in that field.

2. What is Symbolic Logic?

We are now ready to try to answer the question, "What is symbolic logic?" *Symbolic logic* in its broadest sense is a science that has the following characteristics:

(a) It studies mainly non-numerical relations.
(b) It seeks precise meanings and necessary consequences.
(c) Its chief instrument is efficient symbols.

When dealing with one particular field of application, symbolic logic studies the non-numerical statements and relations in that field. When not dealing with any particular field of application, symbolic logic studies the general properties of statements and relations, the foundations of mathematics, and the grounds for reasoning in general.

Other names besides "symbolic logic" have been used for this subject. The other names include: "mathematical logic, axiomatic method, logistic, logistic method". But "symbolic logic" is the name most widely used at present.

3. Comparison of Symbolic Logic and Mathematics

The closest cousin of symbolic logic among the sciences is mathematics. In fact, many people include symbolic logic as part of mathematics. Yet symbolic logic differs from mathematics in a number of ways.

None of the territory of symbolic logic ordinarily includes numbers or numerical ideas like "two, three", or numerical operators like "plus, minus, times, divided by" or numerical relationships like "greater than, less than" or indefinite numbers like "several, most, much". These ideas are all properly part of mathematics.

Mathematics deals mainly with:
numbers, like 3, 1/5, .216
shapes, like a circle or a square
arrangements, like the six possible sequences of the letters A, E, T
patterns, like those in a tiled floor

Symbolic logic deals with:
statements like "Switch *A* is set at position *P*"
classes like "switches, relays, contacts"
relations like "Switch *A* is on only if Switch *B* is on"
properties like "slow-acting, conducting, magnetic"

Mathematics concentrates on answers to questions like: "How much?" "How many?" "How far?" "How long?" Symbolic logic deals with

questions like: "What does this mean?" "Does this set of statements have conflicts or loopholes?" "What is the basis of this proof?"

An example of a rule in mathematics is, "The reciprocal of the reciprocal of a number is the number itself." An example of a rule in symbolic logic is, "The denial of the denial of a statement is the statement itself."

Historically, symbolic logic is the result of applying the powerful technique of mathematical symbolism to the subject matter of logic.

4. A Simple Example of Symbolic Logic

A rather simple example of symbolic logic is the following system of abbreviations for the presence or absence of properties. Suppose we have ten properties each of which may be present or absent in any case. The properties might be ten abilities of people; or ten characteristics of jobs; or ten features in any classification of cases where considerable overlapping of the features may occur. We can set up the following system of abbreviations:

a. For the ten properties, we use the letters *A, B, C, ... , J,* respectively.

b. For any combination of properties present in a case, we use the letters of properties present. (They may be written for convenience in alphabetical order).

c. For the absence of all the properties, we use the letter *Z.*

This system of abbreviation can be useful; each abbreviation tells precisely which properties are present and which are absent; in addition, if we are given some combinations of properties and a rule governing selection from these combinations, then we can promptly write down the combination determined by the rule. For instance, if we have four combinations of properties:

ABCDE, DHIJ, ABDEFHIJ, BCEFHI

and the rule:

> Select that which is present in the first or the second, and is absent in what is common to the third and the fourth,

then the answer is *ACDJ.* For what is present in the first or second is *ABCDEHIJ,* and what is common to the third and fourth is *BEFHI,* and if we exclude the latter from the former, we have *ACDJ* left. The system of abbreviation is not numerical, and it is efficient, and it does permit calculation.

An extension of this system is used in chemistry. The abbreviation NaOH for the compound sodium hydroxide tells that the elements Na

(sodium), O (oxygen), and H (hydrogen) only are present (in the proportions of one atom of each for each molecule of the compound). The abbreviation H_2O for water tells that hydrogen and oxygen are present (in the proportion of two atoms of hydrogen and one atom of oxygen for each molecule of the compound). Here, the facts belong to the science of chemistry, the numbers belong to mathematics, but the system of abbreviation belongs to symbolic logic.

5. The Branches of Symbolic Logic

There are at least four fairly well-recognized branches of symbolic logic. One of these—and the most important branch in practice today—is Boolean algebra, the algebra of AND, OR, NOT, and statements (or classes). For example, a rule from Boolean algebra is that "neither *a* nor *b* is the same as not-*a* and not-*b*." Here *a* and *b* are statements or classes or circuit-elements but not numbers (beyond 1 and 0). Boolean algebra has proved to be useful in designing and checking electrical circuits using relays, electronic tubes, or other on-off circuit elements. This application of symbolic logic is important in the design and construction of intelligent machines, and is described at length in later chapters.

Another branch of symbolic logic is the one that deals with the foundations of mathematics. It has studied such questions as: "What is a number?" "What is a mathematical function?" It has answered these questions to a large extent. One of the great books in the development of symbolic logic is *Principia Mathematica,* by Bertrand Russell and A. N. Whitehead (published Cambridge, England, 1910-13), which to a large extent furnished a logical foundation for all of mathematics.

A third branch of symbolic logic is called the algebra of relations. This deals with such concepts as symmetric relations, transitive relations, connected relations, series, etc.

Still another branch of symbolic logic deals with what are called *decision problems,* that is, finding effective computational procedures for deciding the truth or falsity of whole classes of statements. Symbolic logicians have investigated the problem of proving statements in any mathematical system. These studies have produced some remarkable results. For example, it can be shown that there are classes of statements in arithmetic, and in other mathematical systems, that can never be decided as true or false, in this sense: it can be shown that there exists no effective computational procedure for deciding them.*

*See "Computability and Unsolvability" by Martin Davis, McGraw-Hill Book Co., New York, 1958.

Chapter 2

SYMBOLIC LOGIC: THE BASIC IDEAS

1. An Illustrative Context

To make clear the specific ideas of symbolic logic, and show how they apply and how powerful they are, we shall choose an illustrative context, which is not mathematical, and which in fact belongs in the common everyday experience of all people: the context of family relationships. Within this context we can show how these relationships can be described using two kinds of words: words which belong specifically to the field of family relationships (we can call them "brick" words); and words which are completely general and belong in symbolic logic (we can call them "cement" words).

Problem 1: Define "father" in terms of "parent" and "male", and separate between the brick words and cement words.

Solution: Definition: A person is the *father* of another person if and only if he is male and is the parent of that other person.

Separating the two kinds of words, we have:

(a) Words which belong in the field of family relationships (brick words): person, father, male, parent;

(b) Words which belong in the field of symbolic logic (cement words): A .... is the .... of another .... if and only if it (we replace "he" by "it" to avoid the idea of animateness) is .... and is the .... of that other .....

Problem 2: Express "uncle" in terms of "parent" and "male".

Solution: First, we shall define "brother".

Definition: A person is a brother of another person if and only if he is male and there is somebody who is a parent of both of them.

Brick words (words which belong to family relationships): person, brother, male, body, parent.

Cement words (words which belong to symbolic logic): A .... is a .... of another .... if and only if it is .... and there is some .... which is a .... of both of them. (We must change "who" to "which" so as not to imply animateness.)

Second, having defined "brother", let us define "uncle".

Definition: A person is an uncle of another person if and only if he is a brother of a parent of that other person.

Brick words: person, uncle, male, brother, parent.

Cement words: A.... is a.... of another.... if and only if it is a.... of a.... of that other.....

2. Things, or Elements

Now, we need to get rid of some of the shapelessness of these patterns of logical words, because of their blanks. We need to take out the blanks and put in the most general ideas that are implied by the blanks.

To help us in this process, and enable us to imagine the ideas a little better, let us use "thing" instead of "person". (Actually "entity" or "element" or "individual" would be a better word, in the sense of being more correct technically, because we intend to designate non-physical things also; but "thing" is a comfortable everyday kind of word, and is not an unreasonable choice.)

Also, let us use capital letters *F*, *M*, *P*, *B*, *U* to take the place of words "father, male, parent, brother, uncle", while we try to imagine the kinds of still more general ideas that are needed in place of these letters.

With these changes, our three statements become:

Statement 1: A thing is the *F* of another thing if and only if it (the first-mentioned thing) is *M*, and is a *P* of that other thing (the second-mentioned thing).

Statement 2: A thing is a *B* of another thing if and only if it is *M* and there exists something which is a *P* of both of them.

Statement 3: A thing is a *U* of another thing if and only if it is a *B* of a *P* of that other thing.

We can now begin to see some of the basic ideas of symbolic logic emerging. What these are we shall now try to explain.

3. A ...., Another ...., Still Another ....

In order to talk about the things in some class or collection, we often use some scheme for identifying them one after another, naming them off, and referring to them briefly. This is a process or scheme which belongs to the field of symbolic logic. The need for brief reference to previously mentioned things is recognized in the grammar of natural language, where the *pronouns* "he, she, it, they" (or similar words in other languages than English), are used to refer to previously mentioned things. But in English, the grammar of natural language mixes up thoroughly the logical need for clear designation of previously mentioned things with an unrelated need to specify sex (masculine, feminine, common, or neuter) and number (singular, plural).

In the context of symbolic logic, we are not concerned with sex, grammatical or otherwise. Also, in the context of symbolic logic, we are ordinarily only slightly interested in the difference between singular and plural, although every now and then we find it necessary or desirable in symbolic logic to distinguish between one and more than one.

The translation of "a .... , another .... , still another ...." into symbolic logic can be done in several ways. One way, which is ordinarily not useful, is to use different letters: as in: "a thing x, another thing y, still another thing z". A second way is to use subscripts as in: "a thing x_1, another thing x_2, still another thing x_3, x_4, x_5, ... " A third way is to use no mark for the first-mentioned thing, a prime ($'$) for the second-mentioned thing, a second mark ($''$) for the third-mentioned thing: as in "x, x' (read "x prime"), x'' (read "x second"). If in a discussion, we do not talk of more than three things in a class, this method is rather convenient.

Note that in all these cases these symbols may include the case where another thing x' turns out to be the same as a first thing x; for instance, there will be some occasions where a rule becomes much more general because this case $x = x'$ is not prohibited by the scheme of designating.

4. Properties, Characteristics, Kinds, Sorts

In regard to anything that we talk about, we often make statements telling what properties it has, what characteristics it has, what kind of a thing it is, what sort of a thing it is. For example, "it is male", "it is a father", "it is an uncle", "it is savage", "it is a savage". Note how a property may be grammatically a noun or an adjective.

For designating a property or a class, we shall in general use the capital letter K (the first letter of "kind" and of the German word "Klasse" for class).

$x\,K$ stands for "x has the property K".

$x \epsilon K$ stands for "x is in the class K", "x is a member of K".

Some of the time we think of the property K "being male, having maleness, being a male, being a father, being in fatherhood". At other times we think of the class K "males", "fathers", the group or collection or set of all things which have a stated property. Although the two ideas of property or class are in many ways the same, $x\,K$ and $x \epsilon K$ are two different but largely interchangeable ways used in symbolic logic for expressing this same basic idea.

5. Relations

Also, in regard to anything that we talk about, we often make statements telling what relations it has to other things, what connections or

associations it has with other things. For example: x is an F of x'; x is a P of x'; x is a brother of x'; x has the same father as x'. Any statement which mentions two things and asserts some kind of association or connection between them is a statement expressing a relation.

For designating a relation, we shall in general use the capital letter R (the first letter of the word "relation").

$$x\,R\,x' \text{ means "}x \text{ has the relation } R \text{ to } x'\text{"}$$

We can think of x,x' (read "x comma x prime") as constituting an *ordered pair,* or *ordered couple,* or *dyad,* and then say, if we wish, $x,x' \in R$. This means "the ordered pair x,x' is a member of R". On such occasions we think of the relation R as a class of ordered pairs of elements.

For example, suppose we think of the relation "opposite to" and the four points of the compass "north (N), east (E), south (S), west (W)".

The class of N, E, S, W, standing for the four points of the compass contains four elements. From these we can make 16 ordered pairs:

N, N	E, N	S, N	W, N
N, E	E, E	S, E	W, E
N, S	E, S	S, S	W, S
N, W	E, W	S, W	W, W

Of these 16 only four are in the relation "opposite to", namely:

N, S S, N E, W W, E

The other 12 ordered pairs are in the relation "not opposite to."

For a relation to be completely stated, more than two terms may have to be mentioned. For example, betweenness is a relation with three terms: "New Haven is between Boston and New York." Betweenness is a class of *ordered triples,* a *triadic* relation. For another example, exchange is a relation of four terms: "exchange by somebody of something with somebody else for something else." Exchange is therefore essentially a class of *ordered quadruples,* ordered sets of four elements, a *tetradic* relation.

6. Statements, Sentences, Assertions, Propositions

One of the most basic ideas of symbolic logic is the idea of a statement (sentence, assertion, proposition). In general a statement is an expression which can be true or false. For example: "Jack has had measles", "x is male", "the connection between points A and B is broken".

For designating a statement we shall in general use the letter S.

A statement has a truth value, which is "true" or "yes" or checkmark (v) or T if it is true, and "false" or "no" or cross (×) or F or 0 if it is

false. We write:

$T(S)$ = the truth value of the statement S, which is 1 if S is true, 0 if S is false.

7. The Connectives of Statements

We can easily make new statements from given statements by means of logical connectives (or conjunctions or operators). For example, we can combine two statements with "or" in the sense "and/or". We can also turn a statement into its negative.

$S \cdot S'$	S and S'; both S and S'
$S \vee S'$	S and S' or both; S and/or S'
$S \longrightarrow S'$	if S, then S'; S implies S'
$S \longleftrightarrow S'$	S if and only if S'; if S, then S', and if S', then S

In this collection we also need to put "not", expressing the denial of a whole statement:

$\sim S$	not-S; it is not true that S

There are connectives between statements which are not in the territory of symbolic logic: for example, "S because S'", because cause and effect is in the realm of science; "S while S'", because the idea of time is outside of the field of symbolic logic.

These five operators AND, OR, IF .... THEN, IF AND ONLY IF, NOT all have a very important property: the truth value of a combination statement using them can be calculated from the truth values of the constituent statements. For example, the truth value of S and S' is specified in the inventory of the four possible cases shown in the following table.

One Statement S	Another Statement S'	Statement Resulting from Combining Them with AND $S \cdot S'$
0	0	0
0	1	0
1	0	0
1	1	1

Many connectives are equivalent to combinations of these five connectives. For example, "S unless S'" is the same as "if not-S', then S". For example, "John will come unless it rains" is the same as "if it does not rain, John will come."

Another connective between statements which is in symbolic logic is equality or interchangeability, $S = S'$. This is true if and only if each statement makes the same assertion, as for example, "two plus three make five" and "deux et trois font cinq"; both make the same assertion. But we cannot calculate the truth value of $S = S'$ from knowing the truth valuc of S and the truth value of S'.

8. The All Operator and the Existence Operator

If we have a statement "x has the property K", we can construct another statement:

every x has the property K

This statement is translated into symbolic logic using what is called an "all-operator", and the form in which the statement is then written is:

$$(A\ x)\ (x\ K)$$

which is read, "for every x, x has the property K", or "for all x's, x has the property K".

From the statement "x has the property K", we can also construct the statement:

some x has the property K

This statement has the same meaning as "there exists an x such that x has the property K". These statements are translated into symbolic logic using what is called the "existence-operator", and the form in which they are written is:

$$(E\ x)\ (x\ K)$$

which is read "there exists an x such that x has the property K"

Different conventions are used by different authors writing in the field of symbolic logic for designating these operators. For example, the A of the all-operator may be omitted, so that the expression appears as (x) $(x\ K)$. Or the existence operator may be written with a capital E backwards, a mirror image of the usual capital E. The meaning however is the same even though different symbolic conventions are used.

9. The Three Answer Statements Rewritten in the Symbolic Language of Symbolic Logic

Let us return now to the three statements which were the answers to the two problems given above. How do they look with the new symbols which we have defined?

Statement 1: A thing is the F of another thing if and only if it is M and is a P of that other thing.

$$x\,F\,x' \longleftrightarrow (x\,M)\cdot(x\,P\,x')$$

Statement 2: A thing is a B of another thing if and only if it is M and there exists something which is a P of both of them.

$$x\,B\,x' \longleftrightarrow [x\,M\cdot(\mathrm{E}\,x'')\,(x''\,P\,x\cdot x''\,P\,x')]$$

Statement 3: A thing is a U of another thing if and only if it is a B of a P of that other thing.

$$x\,U\,x' \longleftrightarrow [(Ex'')(x\,B\,x''\cdot x''\,P\,x')]$$

We have now stripped away the English language for these statements, and expressed the exact skeleton of the relationship in the symbols of symbolic logic, which calculate. This is a worthwhile achievement, because we have escaped the various paraphrases of language, we have crystallized out the ideas, we have prepared ourselves for calculation and exact reasoning. Over and over again, in the progress of a mathematical science or art, the capacity to free oneself from the accidental or irrelevant, and concentrate on the fundamental and relevant, has been the gateway to progress.

10. The "Class of ..." Operator

If we have the statement "x has the property K", we can talk of:

the class of all x's that have the property K

This class is designated as:

$$(C\,x)(x\,K)$$

read "the class of all x's such that x has the property K". This class is equal to K, in the sense of class rather than property.

11. The Relative Product

If:

x is an R of an R' of x'

or, in other words:

x has the relation R to something which has the relation R' to x'

we write:

$$x\,R\,|\,R'\,x'$$

The exact meaning of this is:

$$x\,R \mid R'\,x' = (Ex'')(x\,R\,x'' \cdot x''\,R'\,x')$$

$R \mid R'$ is spoken of as the *relative product* of R and R'.

With these definitions, we are now ready for a tougher problem.

12. Definitions of Family Relationships

Problem 3: In describing the relationships of human beings, the follow-ideas may be used:

1. mother	6. aunt	11. siblings	16. mother-in-law
2. child	7. grandparent	12. first cousin	17. illegitimate child
3. sister	8. husband	13. half-brother	18. ancestor
4. son	9. wives	14. great-aunt	
5. daughter	10. spouse	15. nephew	

(a) What is a list of three basic terms in the field of family relationships, which, together with the ideas of symbolic logic, will express every one of these ideas?

(b) What is their expression?

(c) How can a man be his own grandfather-in-law?

Solution: (a) A list of three basic terms is "Male, Parent, Wedded". (Other lists of three basic terms may be used instead, such as "Female, Child, Wedded".) Let

$$xM = x \text{ is Male}$$
$$xPx' = x \text{ is a Parent of } x'$$
$$xWx' = x \text{ is Wedded to } x'$$

(b) Using the symbols of symbolic logic, we can express all 18 ideas given in the problem with compact and precise definitions as follows:

1. x Mother $x' = xPx' \cdot \sim xM$

In words: "x is the mother of x'" is the same as
"x is a parent of x prime, and x is not male."

2. x Child $x' = x'Px$

In words: "x is a child of x'" is the same as
"x prime is a parent of x"

3. x Sister $x' = \sim xM \cdot (Ex'')(x''Px \cdot x''Px')$

In words: x is not male, and there exists another person, x second, such that x second is a parent of x, and x second is a parent of x prime.

4. x Son $x' = x'Px \cdot xM$

x prime is a parent of x, and x is male.

5. x Daughter $x' = x'Px \cdot \sim xM$
   x prime is a parent of x, and x is not male.
6. x Aunt $x' = x$ Sister $|$ Parent x'
   x is a sister of a parent of x prime.
7. x Grandparent $x' = x$ Parent $|$ Parent x'
   x is a parent of a parent of x prime.
8. x Husband $x' = xWx' \cdot xM$
   x is wedded to x prime, and x is male.
9. Wives $x' = (Cx)(xWx' \cdot \sim xM)$

In words: the class of all x's such that x is wedded to x prime, and x is not male.

10. x Spouse $x' = xWx'$
    x is wedded to x prime.
11. Siblings $x' = (Cx)\,[x(\text{Brother v Sister})x']$
    the class of all x's such that x is a brother or a sister of x prime.
12. x First Cousin $x' = x$ Child $|$ (Brother v Sister) $|$ Parent x'
    x is a child of a brother or a sister of a parent of x prime.
13. x Half-Brother $x' = xM \cdot E(x_2, x_3)(x_2 \neq x_3 \cdot x_2Px \cdot x_2Px' \cdot x_3Px \cdot \sim x_3Px')$
    x is male, and there exist two different persons, x two and x three, such that x two is a parent of x, and x two is a parent of x prime, and x three is a parent of x, but x three is not a parent of x prime.
14. x Great-Aunt $x' = x$ Sister $|$ Grandparent x'
    x is a sister of a grandparent of x prime.
15. x Nephew $x' = x$ Son $|$ (Brother v Sister) x'
    x is a son of a brother or a sister of x prime.
16. x Mother-in-Law $x' = x$ Mother $|$ Spouse x'
    x is mother of a spouse of x prime.
17. x Illegitimate Child $x' = x'Px \cdot [(Ex'')(x'' \neq x' \cdot x''Px \cdot \sim x''Wx')]$
    x prime is a parent of x, and there exists an x second such that x second is different from x prime, and x second is a parent of x, and x second is not wedded to x prime.
18. x Ancestor $x' = (EK)\,\{(x \in K \cdot x' \in K) \cdot (Ax_2)[(x_2 \in K \cdot x_2 \neq x) \rightarrow (Ex_3)(x_3Px_2 \cdot x_3 \in K)]\}$

In words: x is an ancester of x' if and only if there exists a class of persons K (including x and x') which has this property: for every person x_2 in K (except for x) there exists some person x_3 in K such that x_3 is the parent of x_2.

(c) A man a marries the mother m of a girl g who marries the man's father f. That is: a Spouse $m \cdot m$ Parent $g \cdot g$ Spouse $f \cdot f$ Parent a. Therefore:

$$a \text{ Spouse} \mid \text{Parent} \mid \text{Spouse} \mid \text{Parent } a$$

and so a is his own grandfather-in-law.

13. The Main Concepts of Symbolic Logic

Exactly what are the important ideas in the mean stream of symbolic logic, and what are the symbols used for them?

Many of the main ideas of symbolic logic are defined and explained in "Principia Mathematica" mentioned above, although in that work they are very hard to understand, because of the effort made to escape preconceived ideas by using solely a symbolic language. A second good source to find these ideas is in Chapter 2 of "The Axiomatic Method in Biology".*

Most of the concepts from this main stream of symbolic logic, with some modifications due to later investigators, are given in the following brief, condensed summary. In this summary, some difficult typographic symbols have been replaced by simpler ones. This summary leads up to the definitions of *cardinal number*, the whole number that counts the number of elements in a class, and *series*, the mathematical concept of a set of elements having a serial order, that can be arranged one after another in a single sequence.

It is not easy nor desirable to try to understand these ideas by just reading them over one after another. One needs to get used to them over a long time, and by reading and studying books on symbolic logic, such as W. V. Quine's "Mathematical Logic".

Also, many of these ideas are not yet necessary for the present-day applications of symbolic logic to intelligent machines. For such applications have so far made use only of Boolean algebra, the algebra of AND, OR, NOT, and conditions (see later chapters). But nevertheless Boolean algebra is only a small part of symbolic logic; and probably much more of symbolic logic will eventually come out of the realm of pure and unapplied disciplines, and be applied in the analysis and programming of intelligent machines.

---

*J. H. Woodger, University Press, Cambridge, Eng., 1937, 174 pp.

TABLE 2-1. THE IMPORTANT IDEAS OF SYMBOLIC LOGIC

Symbol	Meaning	Definition
$x, x', x'', \ldots.$	things, individuals, ... (read "x, x prime, x second, ..." standing for "x, another x, still another x, ...")	primitive
$K, K', K'', \ldots.$	classes, groups, sets (K, first letter of German "Klasse" meaning "class")	primitive
$R, R', R'', \ldots.$	relations (R, first letter of "relation") *Note:* These are nearly always two-termed relations	primitive
$S, S', S'', \ldots.$	statements, propositions (S, first letter of "statement")	primitive
$\sim S$	not-S	primitive
$S \vee S'$	S or S' or both	$\sim(\sim S \cdot \sim S')$
$S \cdot S'$	S and S'	primitive
$S \rightarrow S'$	If S, then S'; S implies S'	$\sim(S \cdot \sim S')$
$S \leftrightarrow S'$	S if and only if S'	$(S \rightarrow S') \cdot (S' \rightarrow S)$
ϵ	is an element of, is a member of, is in (epsilon, the first letter of Greek "esti", meaning "is")	primitive
$x \epsilon K$	The thing x is in the class K	primitive
xK	The thing x has the property K (*Note:* This form is seldom used)	equivalent in meaning to $x \epsilon K$
xRx'	x has the relation R to x'	primitive
$x, x' \epsilon R$	The ordered pair of things x, x' is in the relation R (*Note:* This form is seldom used)	equivalent in meaning to xRx'
$(Ax)(x \epsilon K)$	For every x, x is in K. Every x is a K. (applies for a range of admissible values of x) (A, first letter of "all")	$\sim(Ex)(\sim x \epsilon K)$

TABLE 2-1. (*Continued*)

Symbol	Meaning	Definition
$(Ex)\ (x \in K)$	There exists an x such that x is in K. Some x is in K. (E, first letter of "exists")	primitive
$x = x'$	x equals x'	$(AK)\ (x \in K \leftrightarrow x' \in K)$
$(Cx)\ (x \in K)$	the class of all x's such that x is in K (applies for a range of admissible values of x) (C, first letter of "class")	primitive
$(tx)\ (x \in K)$	the x such that x is in K (t, the first letter of "the")	primitive, but it is necessary that (Ex) $[(Ax')\ (x' \in K \rightarrow x' = x)]$
K	the class K; the class of all x's such that x is in K	$(Cx)\ (x \in K)$
$K \subset K'$	K lies in K'	$(Ax)\ (x \in K \rightarrow x \in K')$
$K = K'$	K equals K'	$(Ax)\ (x \in K \leftrightarrow x \in K')$
$K \vee K'$	K or K' or both	$(Cx)\ (x \in K \vee x \in K')$
$K \cdot K'$	K and K'	$(Cx)\ (x \in K \cdot x \in K')$
$\sim K$	not-K	$(Cx)(\sim x \in K)$
V	universal class	$(Cx)(x \in K \vee \sim x \in K)$
Λ	null class	$(Cx)(x \in K \cdot \sim x \in K)$
$K \neq K'$	K and K' are distinct	$(Ex)\ [(x \in K \cdot \sim x \in K') \vee (x \in K' \cdot \sim x \in K)]$
R	the relation R; the class of all ordered pairs x, x' such that xRx'	$(Cx, x')\ (xRx')$
$R \subset R'$	R lies in R'	$(Ax, x')\ (xRx' \rightarrow xR'x')$
$R = R'$	R equals R'	$(Ax, x')(xRx' \leftrightarrow xR'x')$
$R \cdot R'$	the logical product of the two relations R and R'	$(Cx, x')\ (xRx' \cdot xR'x')$
$R \vee R'$	the logical sum of the two relations R and R'	$(Cx, x')\ (xRx' \vee xR'x')$
$\sim R$	the negative of the relation R; the class of all pairs x, x' such that xRx' is false	$(Cx, x')\ (\sim xRx')$

TABLE 2-1. (*Continued*)

Symbol	Meaning	Definition
$R`x'$	the R of x'; the x such that xRx'	$(\iota x)\ (xRx')$
$\overrightarrow{R}`x'$	the R's of x'; the class of all x such that xRx'	$(Cx)\ (xRx')$
$\overleftarrow{R}`x$	the class of all x' such that xRx'	$(Cx')\ (xRx')$
$R``K$	the R's of K's	$(Cx)\ [(Ex')\ (x' \in K \cdot xRx')]$
$D`R$	the domain of R	$(Cx)\ [(Ex')\ (xRx')]$
$\breve{C}`R$	the converse domain of R	$(Cx')\ [Ex)\ (xRx')]$
$F`R$	the field of R	$(Cx)\ (Ex')\ (xRx' \vee x'Rx)$, which reduces to $D`R \vee \breve{C}`R$
$\breve{R}$	the converse of a relation R; the class of ordered pairs x', x such that xRx'	$(Cx', x)\ (xRx')$
$R \mid R'$	the relative product of R and R'	$(Cx, x'')\ [(Ex')\ (xRx' \cdot x'R'x'')]$
R^2	the square of a relation R	$R \mid R$
R^3	the cube of a relation R	$R^2 \mid R$
sym	the class of symmetric relations	$(CR)\ (Ax, x')\ (xRx' \rightarrow x'Rx)$
asm	the class of asymmetric relations	$(CR)\ (Ax, x')\ (xRx' \rightarrow \sim x'Rx)$
refl	the class of reflexive relations	$(CR)\ (Ax, (x \in F`R \rightarrow xRx)$
irr	the class of irreflexive relations	$(CR)\ (Ax, x')\ (xRx' \rightarrow x \neq x')$
trans	the class of transitive relations	$(CR)\ (Ax, x', x'')\ (xRx' \cdot x'Rx'' \rightarrow xRx'')$
connex	the class of connected relations	$(CR)\ (Ax, x')\ (x, x' \in F`R \rightarrow (xRx' \vee x'Rx \vee x = x')$
intr	the class of intransitive relations	$(CR)\ (Ax, x', x'')\ (xRx' \cdot x'Rx'' \rightarrow \sim xRx'')$
onma	the class of one-many relations	$(CR)\ (Ax, x', x'')\ (xRx'' \cdot x'Rx'' \rightarrow x = x')$
maon	the class of many-one relations	$(CR)\ (Ax, x', x'')\ (xRx' \cdot xRx'' \rightarrow x' = x'')$
onon	the class of one-one relations	onma · maon

TABLE 2-1. (*Continued*)

Symbol	Meaning	Definition
K Sim K'	the class K is similar to class K'	$(\exists R)\ (R \in \text{onon} \cdot \mathrm{D}`R = K \cdot \mathrm{C}`R = K')$
$\mathrm{Nc}`K$	the cardinal number of K	$(\mathrm{C}K')\ (K' \text{ Sim } K)$
ser	class of serial relations	asm · trans · connex

14. Are There Other Ideas that May Be Expressed in Efficient Symbols?

After this look at the basic ideas of symbolic logic it is appropriate to ask several questions:

(1) Are there other ideas in language that can be crystallized out into efficient symbols that can be calculated with?
(2) Is there a possibility that all the language of thought can become calculable like mathematics and symbolic logic?
(3) Is it conceivable that, in the future, arguments of all kinds will be settled not by human beings arguing about them but by automatic computers calculating the answers?

I believe that the answer to all of these questions is "yes."

Ordinary natural language is the matrix, the ore, the molten rock, out of which crystallize the diamond-like ideas and symbols of mathematics and symbolic logic. As human beings more and more understand the natural languages they use, the ways in which these languages reveal and hide ideas, reflect and distort concepts, the more they will manage to extract from languages the crystals and nuggets of clarity that are to be found in such fields as mathematics and symbolic logic. The crucial point, of course, is not making a new symbol—but making useful symbols that can be calculated with, that can be rigorously connected with a host of other symbols in the relations of logic and mathematics.

The history of the development of mathematics and symbolic logic is a clear proof of these views. Ideas that have been unexpressed or badly expressed in language for centuries have gradually come out of language and have been deposited in the fabric of efficient symbols. And the advent of powerful automatic computers will tend to hasten the process, opening great new realms of thought to symbolic calculation. An example of a newly crystallized branch is the algebra of states and events, described in a subsequent chapter of this book.

Chapter 3

BOOLEAN ALGEBRA: INTRODUCTION

1. The Part of Symbolic Logic with Useful Applications

The part of symbolic logic which has so far found many useful and unexpected applications is the technique for handling AND, OR, NOT, and conditions, called *Boolean algebra.*

Some of these applications have been in the field of rules, contracts, and agreements, for example, classifications of cases and the corresponding actions to be taken with regard to them. Here a typical problem to which Boolean algebra applies is making sure that sets of statements have no conflicts and no loopholes. Here, the kind of Boolean algebra is the *algebra of classes*—a technique for symbolic calculating about classes or collections of cases, with the connectives AND, OR, NOT, EXCEPT, etc.

But most of the important applications of Boolean Algebra have been in the field of switches, circuits, signals, controlling, and computing—various parts of the field of the design and construction of intelligent machines. Here Boolean algebra is the *algebra of "on-off" circuit elements*—a technique for symbolic calculating about combinations of elements of circuits which may be either on or off, conducting or not conducting, in one position or the other position, etc., combined with physical connections which express AND, OR, NOT, EXCEPT, etc.

How can Boolean algebra apply to two such diverse fields? The reason essentially is that this algebra handles the ideas expressed in many of the commonest words of language. So it is reasonable to expect that Boolean algebra will apply in many fields, because these words are used in many fields.

2. Words Belonging to Boolean Algebra

One way to gain an idea of the content of Boolean algebra is to notice the words of ordinary language that are in the territory of Boolean algebra, in the same way that PLUS, MINUS, TIMES, DIVIDED BY are in the territory of ordinary algebra. These words are shown in Table 3-1.

Most of the meaning of these words is dealt with exactly and precisely in Boolean algebra. But not all the meaning, since words are products of

the evolution of language, and a word often has meanings from different territories of thought fused together within it. Take for example the word "but" when connecting two statements. "But" and "and" when connecting two statements are translated indistinguishably in Boolean algebra as AND, because they connect two statements to assert that they are both true. In other words, "but" has the same logical meaning of association as the word "and"; however, as compared with "and", "but" conveys an additional idea, "you would not have expected it", "contrary to expectation". And variations of expectation expressed by the speaker to the listener, are not relevant in Boolean algebra.

If we compare these words with the list of words in Table 1-1, we shall not find much overlapping. The two tables together show a good many of the words that belong to symbolic logic; the first table shows some words which are in symbolic logic but not in Boolean algebra.

TABLE 3-1. COMMON WORDS AND PHRASES BELONGING TO BOOLEAN ALGEBRA

a
all
an
and
and/or
any
anything
are

be
both
but

class
collection
consists of
contains

each
either
either .... or
empty
everything
except
excludes
excluding

group (meaning "class")

if
if and only if
if ...., then
in
includes
including
is
is contained in
is included in

kind (meaning "class")

lies in

neither
neither .... nor
no
none
nor
not
not both
nothing

only
only if
overlaps
or
or .... or both
or .... or .... or any one or more of them
or .... but not both
or else
overlaps

.... s (meaning "plural")
set (meaning "class")
sort (meaning "class")
some
something

type (meaning "class")

unless

which is
which are

yes

3. A Type of Problem and Answer in Boolean Algebra

Another good way to gain an idea of what Boolean algebra is about is to read a typical problem and its answer. Here is such a problem, due to John Venn, 1888.

Problem: A certain club has the following rules: (1) The financial committee shall be chosen from among the general committee. (2) No one shall be a member of both the general and the library committees unless he is also on the financial committee. (3) No member of the library committee shall be on the financial committee. Simplify these rules.

Answer: The rules may be simplified as follows: (1) The financial committee shall be chosen from among the general committee. (2) No member of the general committee shall be on the library committee.

What is important here is the fact, here reported, that (1) three statements with seven mentions of classes are reduced to two statements with four mentions of classes; and (2) the simplified rules in the answer can be rigorously proved to be exactly equivalent to the longer rules in the problem. A way in which this answer can be calculated using Boolean algebra may be found on page 45.

And in general we can, with Boolean algebra, take sets of rules or conditions and manipulate them into various forms that are logically equivalent or are logically implied, and we can do this directly, rigorously, and easily. Also, if the amount of calculation with Boolean algebra is too much or too arduous for a human being, it is quite possible to put the calculation on an automatic computing machine, and in this way obtain the answer.

4. The Origin of Boolean Algebra

Boolean algebra is named after a great English mathematician, George Boole, who lived 1815 to 1864, and who in 1854 published "The Investigation of the Laws of Thought", a book which has become famous. At the start of Chapter I, he wrote:

"The design of the following treatise is to investigate the fundamental laws of those operations of the mind by which reasoning is performed; to give expression to them in the symbolical language of a Calculus; and upon this foundation to establish the science of logic and construct its method; to make that method itself the basis of a general application of a mathematical doctrine of the Probabilities; and finally to collect from the various elements of truth brought to view in the course of these inquiries some probable intimations concerning the nature and the constitution of the human mind."

In this book Boole laid out by far the largest part of a new algebra, particularly suited to handling classes and propositions (statements).

Other mathematicians and logicians, including John Venn and Ernst Schröder, subsequently greatly improved and extended Boole's algebra.

In 1938 Claude E. Shannon, then a student at Massachusetts Institute of Technology, afterwards a well-known mathematician and engineer at Bell Telephone Laboratories, and at present writing a member of the faculty at M.I.T., showed that Boolean algebra could be applied excellently in the design of on-off circuits. His paper, "A Symbolic Analysis of Relay and Switching Circuits"* is a milestone in the development of applications of Boolean algebra.

5. How Does Boolean Algebra Compare with Ordinary Algebra?

The ordinary algebra learned in high school and college deals with:

- numbers, and letters that stand for them—*elements;*
- *operations*, such as addition, multiplication, taking to a power, etc.; and
- other ideas, like equations, functions, unknowns, etc.

This algebra has acquired the specific name *elementary algebra*, although many of its problems are in no way elementary, because mathematicians had to have some name for it to distinguish it from other recently developed modern algebras, which bear names like linear associative algebra, the algebra of lattices, the algebra of rings, etc.

Boolean algebra is just as much an algebra as is elementary algebra, although it deals with different elements and different operations. In many ways it resembles elementary algebra, but in quite a few ways it is simpler and easier.

Elementary algebra chiefly applies to numbers, including fractions and roots. Boolean algebra chiefly applies to classes, statements, and conditions. The classes may be classes of employees, classes of computing machines, classes of contracts,—in fact, all classes, types, sorts, or collections.

Elementary algebra operates with PLUS, MINUS, TIMES, DIVIDED BY, and some other operations. Boolean algebra operates with OR, AND, NOT, and EXCEPT, in almost their regular meanings.

Elementary algebra has a rule $a + a = 2a$, a PLUS $a = 2a$. This is true whatever number may be represented by a. In Boolean algebra the corresponding rule is $a \vee a = a$, or a OR $a = a$, true whatever class may be represented by a. For the vee sign (v) stands for OR, the OR that does

*Thesis for the Master's Degree at M.I.T., Published by the American Institute of Electrical Engineers.

not exclude both alternatives, ... OR .... OR BOTH, also expressed AND/OR; and on the basis of all our experience, *a* OR *a* OR BOTH is just another and longer way of saying *a*. An example of the elementary algebra rule is 7 PLUS 7 = TWICE 7. An example of the Boolean algebra rule is: employees age 50 or over, OR employees at least age 50, OR BOTH = employees who have passed their 50th birthday. (The words describing the class make no difference, provided the contents of the class is unchanged.)

The "v" is selected to stand for "OR" because it is the initial letter of the Latin word "vel" meaning "and/or", in contrast to the Latin word "aut" meaning "or else". It is interesting to note that the Romans distinguished "and/or" from "or else" by two different words.

In elementary algebra $a \cdot a = a^2$, *a* TIMES *a* = *a* SQUARED. But in Boolean algebra $a \cdot a = a$, *a* AND *a* = *a*. For the centered dot (·) now stands for .... AND .... , in the sense of BOTH .... AND .... ; all our experience confirms that BOTH *a* AND *a* is the same as *a* alone.

Here therefore is one way in which Boolean algebra is simpler and easier than elementary algebra: no numbers appear in Boolean algebra. For $a \vee a$ or $a \vee a \vee a$ or $a \cdot a$ or $a \cdot a \cdot a$ or.... are all simply repetitions of *a*, and may at once be replaced by *a* alone. This means that there are in Boolean algebra no multiples, no fractions, no powers, no roots, no numerical coefficients of any kind. And consequently, calculations in Boolean algebra are often faster and easier.

The third operation in Boolean algebra is NOT. NOT-*a* is written a'. NOT has some of the character of MINUS and some of the character of THE RECIPROCAL OF; for NOT NOT $a = a$, just as $-(-a) = a$ and $1/(1/a) = a$.

No new sign is really needed for EXCEPT, for it may be expressed easily with AND and NOT. The class of things *a* EXCEPT *b* is the same as the class of things which are *a* AND NOT *b*, $= a \cdot b'$.

Nor is a new sign needed for the other OR, OR ELSE, the OR which does exclude both alternatives. For, *a* OR *b* BUT NOT BOTH is the same as *a* OR *b* EXCEPT BOTH *a* AND *b* and so may be expressed $(a \vee b) \cdot (a \cdot b)'$. This expression may be simplified into $a \cdot b' \vee b \cdot a'$.

In other words, Boolean algebra is an algebraic technique for handling classes of things with the operations of AND, OR, and NOT. It seems simple, and it is simple. Elementary algebra is really so complicated that the human race required centuries to develop a sufficiently good symbolic technique to solve the problems about numbers which confronted it. The subject matter of Boolean algebra is, on the other hand, so simple relatively that men have muddled along for centuries without an algebraic expression for the subject matter. They have used instead a

clumsy and very incomplete substitute, the part of formal logic dealing with syllogisms, dating from Aristotle. Because of clumsy tools, mankind has laboriously corrected plenty of logical errors.

6. The Concept of a Class

Now so far we have used the concept of a class, or group, or kind, or sort, or type, or collection, or species, without giving very much attention to it. It is, however, an extremely fundamental idea—fundamental in the same way that the idea of a statement (or proposition or assertion) is fundamental—one of the foundation stones of thought.

The idea of a class, of course, comes from observations of the real world, the scientific world. I can think of no better comment on the idea of class as derived from observations of the real world than the following remarks made by Sir George Paget Thomson, well-known physicist and Nobel prize winner, in his penetrating book "The Foreseeable Future".*

"There is one feature of the world we live in which is so general and so universal that it seems to have escaped proper notice. For want of a better name, I will call it the 'principle of mass production'. It is the tendency which nature shows to repeat almost indefinitely each entity it makes. This is most obvious perhaps among the smallest of objects. There are about enough atoms in the ink that makes one letter of this sentence to provide one not only for every inhabitant of the earth, but one for every creature if each star of our galaxy had a planet as populous as the earth. But there are in the universe less than one hundred kinds of atoms, and these hundred themselves are made of a very small number, two or three, of common constituents; electrons, protons, and neutrons (if the latter are to be given independent status). At this level the individuals of the multitude are identical; that they are strictly indistinguishable is a principle of the quantum theory which might rank with our other principles,.... Larger objects are no longer strictly identical but closely similar. Examples come both from animate and from inanimate nature. Drops of rain, grains of sand, particles of smoke, bacteria, the cells of any piece of apparently homogeneous organic tissue, in every case though there may be a large variety of distinguishable kinds, each kind exists in numbers which even the cold mathematician must describe as considerable and which to the ordinary person are incalculably immense.

"Especially in the living world this is noticeable. A beech tree is one of a species which contains a vast number of individuals, each indeed different, but clearly distinguished from other creatures made of

*Cambridge, Eng.: The University Press, 1955, pp. 9–11):

much the same materials—whales or orchids for example. Each tree has in season a large though perhaps not quite so large number of leaves; each leaf is made up of relatively few kinds of cell, each kind present in very large numbers.

"To my mind this multitudinousness is the outstanding fact in the universe as we know it. There is nothing in logic to demand it. Though apparent to the careful observer using only his unaided senses, the advance of scientific instruments and knowledge makes it much more striking. To a certain extent indeed it is the result of language; we give names to recognizable classes and since we have only a limited number of names, the classification is sometimes arbitrary, giving the impression of sharply defined species where perhaps none exists. But it is more true to say that natural classification has moulded our language. Because there exist recognizable groups each containing many individuals, man has invented the 'universals' which have played such a part in metaphysical argument.

"This, one may be sure is one of the fundamentals of the world which further discovery will not alter. Atomicity in its widest sense, mass production by nature, is the deepest of scientific truths."

Chapter 4

BOOLEAN ALGEBRA: BASIC IDEAS

1. Fundamentals

The fundamental ideas of Boolean algebra (as interpreted in the algebra of classes) are as follows.

(1) We are talking about classes, groups, or sets, which may be labeled with letters $a, b, c, \ldots, x, y, \ldots$.

Why do we now abandon x, x', x'', and $x_1, x_2, x_3, x_4 \ldots$? Because in the context of the Boolean algebra of classes, we have only one kind of mathematical thing to keep track of, not half a dozen kinds as we have in symbolic logic, whose things, properties, classes, relations, statements, etc., all have to be kept track of. So it is convenient to go back to a notation which is more like the notation of elementary algebra.

(2) $a \vee b$ (read "a vee b") stands for a OR b OR BOTH; the class of things contained either in the class a or in the class b or in both classes.

Sometimes in the literature $a + b$ is used for $a \vee b$, but this is not convenient because there is an interpretation of Boolean algebra (which we shall explain later) in which the numbers 1 and 0 are the elements designated by the letters $a, b, c, \ldots$, and it turns out that $a \vee b = a + b - bc$, where PLUS (+) and MINUS (−) now have their ordinary arithmetical meaning; and so it is useful to reserve the plus sign for its regular meaning.

(3) $a \cdot b$, ab (read "a dot b", "ab") stands for a AND b; the class of things contained in both the class a and the class b.

It is convenient to have two signs for AND (dot and juxtaposition) just as in arithmetic the slant (/) and the division sign (÷) both stand for DIVIDED BY. In the interpretation of Boolean algebra in which there are just two numbers 1 and 0 designated by $a, b, c, \ldots$, a AND b with Boolean algebra meaning is the same as a TIMES b with arithmetical meaning, and so the same notation can be used for both of them with no conflict.

(4) a' (read "a prime") stands for NOT-a, which in any discussion is the class of all those things being discussed which are not in the class a.

Sometimes in the literature $\bar{a}$ (read "a dash") is used for NOT-a, but this notation is not convenient in printing and writing, especially when

it is necessary to designate repeated negatives of involved expressions; for example $((a' \vee b)' \vee c)'$ is much more convenient than $\overline{\overline{(a \vee b)} \vee c}$. Sometimes in the literature $\sim a$ is used for NOT-a; this notation is much more convenient than $\bar{a}$ (a-dash), but the tilde ($\sim$) is missing from the typewriter, and it is three unit spaces wide, and so makes expressions longer than they otherwise would be.

(5) Z or 0 (read "nothing, emptiness, null class, zero") stands for the null class; the class which contains nothing; the class which contains no things at all.

Whether or not this class "exists", it is convenient to treat it as existing. There is a difference between the number zero and the null class: the number zero counts the number of things in the null class. In Boolean algebra, most of the time the number zero does not appear at all, and so it is convenient and useful to use the sign 0 (zero) for the null class. In the interpretation of Boolean algebra which consists of the two numbers 1 and 0, what corresponds to the null class is the number 0, and so no confusion arises here either.

(6) U or 1 (read "all, the universe class, one") stands for the universe class; the class of all the things contained in a given discussion (and which therefore changes from one discussion to another, but remains the same in any one discussion). The persons discussing must understand and agree about the universe class; otherwise, NOT has ambiguous meaning, although EXCEPT will still be unambiguous.

There is a difference between the number of elements in the universe class and the number one. In every discussion, the number of elements in the universe class is a positive whole number; often it is a large number or an infinite number, and rarely is just one. Since the number one only occasionally appears in a discussion using Boolean algebra, it is convenient to use the number 1 to designate the universe class. In cases where a confusion may arise, between the universe class and the number one, it is convenient to use the capital letter U to denote the universe class.

(7) $a = b$ (read "a equals b") stands for "a is interchangeable with b"; true if and only if the things contained in the class a are the same things that are contained in the class b.

Here is a place where a distinction between properties and classes appears. One class is the same as another class when it contains exactly the same elements. But one property may not be the same as another property even if the descriptions result in identifying the same class. For example, suppose we are discussing all animals now on earth, and

think of the properties "being a unicorn" and "being a dinosaur". It is true that in this discussion the class of unicorns is empty; and the class of dinosaurs is empty; and the two classes are both equal to the null class and are hence equal to each other. But it is far-fetched and not necessary to assert that the property of being a unicorn is the same as the property of being a dinosaur. It is as if there is a contradiction between the idea of a property and the limitation of "the class of all things being discussed in a given discussion". It is as if the idea of a property actually applies to anything whatever for which it is meaningful to assert ".... has --- property".

(8) $a \neq b$ (read "a is not equal to b") stands for "a is not interchangeable with b"; true if and only if there is at least one thing contained in one of a or b and not contained in the other.

(9) $a \subset b$ (read "a is in b, a horseshoe b") stands for a LIES IN b, a IS CONTAINED IN b; true if and only if all the things contained in the class a are contained in the class b.

Some other signs besides this horseshoe ($\subset$) are sometimes used in the literature for the relation "lies in", "contained in". In the Boolean algebra of statements, for example, the relation is regularly symbolized by an arrow ($\longrightarrow$), but $\longrightarrow$ is less appropriate for classes than $\subset$ in $a \subset b$ because the wider opening of the horseshoe at the right next to b neatly suggests that the more inclusive class is b.

2. Recognition of Boolean Algebra Ideas Expressed in Ordinary English

Now, how do we recognize these fundamental ideas when expressed in ordinary language? There are many different phrasings for them, in addition to the particular phrasings we used above when we defined the fundamental ideas of Boolean algebra. Because of these differences, how to translate from ordinary English into Boolean algebra, is not simple, and requires study.

In the same way, before human beings can use arithmetic, they have to gain experience with particular numbers such as one, two, three, and larger numbers. They must also gain an ability to recognize numbers in the environment around them, and in language, and they need to learn to recognize the relations of numbers. We all start on this task when we are very young; the first number we learn appears to be two; and by the time the first grades of school come along, we have made a practical everyday beginning in arithmetic.

Before we can use Boolean algebra easily, we must gain an experience of what the universe class, the null class, and other classes mean;

we must form a habit of recognizing classes. We must also acquire an ability to see the relations of classes, both in our environment and in language. To gain these capacities is not inherently difficult, but it depends on noticing many instances which show, step by step, the process of recognizing classes and their relations.

So we shall consider many examples of statements and their translations into Boolean algebra, and we shall especially notice the many different paraphrases of ordinary English, each of which expresses one and the same Boolean situation.

3. Class Inclusion

Let us take an example. Suppose we choose the class of all employees of the Great Lakes Company (fictitious) as governed by the statements in the (hypothetical) booklet *Company Rules and Pension Plan Regulations.* This class, then, is the universe class for this discussion.

EXPRESSION 1:

All Great Lakes Company Pension Plan members receive additional retirement income.

Translation:

ALL (Great Lakes Company Pension Plan members) ARE (employees receiving additional retirement income).

All g are d.

g IS CONTAINED IN d.

$g \subset d$.

Paraphrases:

Great Lakes Pension Plan members receive additional retirement income.

The members of the Great Lakes Pension Plan receive additional retirement income.

Each Great Lakes Pension Plan member receives additional retirement income.

Every Great Lakes Pension Plan member receives additional retirement income.

A member of the Great Lakes Pension Plan receives additional retirement income.

Any member of the Great Lakes Pension Plan receives additional retirement income.

The class of employees members of the Great Lakes Pension Plan *is included in* the class of employees who receive additional retirement income.

Notice how the predicate "receive additional retirement income" is paraphrased into a predicate with exactly the same meaning "are employees receiving additional retirement income", a predicate that refers to a class "employees receiving additional retirement income." Another example: The sentence "all mantelops hile" can be converted into "All mantelops are mantelops that hile." "All x's have the property K" can be converted into "All x's are x's that have the property K." Very large numbers of sentences of English can therefore readily be converted into statements expressing relations between classes.

Here we have, first, an expression in ordinary English (in this case, a full statement); secondly, its translation into Boolean algebra (both the steps and the result); and thirdly, some of the ordinary paraphrases of this expression (other ways existing in English of saying the same thing). Every one of these paraphrases is translated into the same result, $g \subset d$. Those words and suffixes in the paraphrases which signal the relation and indicate that the same relation persists have been italicized on account of their importance. This example of a Boolean algebra situation illustrates how this technique for reasoning economizes thought: instead of a dozen ways of saying something, each requiring a dozen words, we have now simply $g \subset d$.

Let us compare this with an example in elementary algebra. Suppose we have the statement:

EXPRESSION:

The quantity of work done by a man in an hour varies directly as his pay per hour and inversely as the square root of the number of hours he works per day.

Translation:

(the quantity of *w*ork done by a man in an hour) VARIES DIRECTLY AS (his *p*ay per hour) AND INVERSELY AS (the square root of) (the *n*umber of hours he works per day)

w varies directly as p and inversely as $\sqrt{n}$

$w \propto p \cdot (1/\sqrt{n})$

Clearly, the two processes of translation have a great deal in common. The translation is not carried out in one mental jump, from words of ordinary English into precise symbols. It is carried out in four steps:

(1) Notice the things mentioned; mentally set them apart, separate them out, group them as a unit, put mathematical parentheses around them.

(2) Choose brief memory-aiding symbols to stand for them.

(3) Then notice the remaining words which express relations.

(4) Identify these relations and give them the symbols regularly used to express them.

In elementary algebra, the things mentioned are nearly always numbers. If the problem tells us exactly what the number is, we most often use its regular name in the decimal notation. For example, if we are told that a number is a "dozen" we put down "12", meaning one ten and two units. If we are not told what the number is, we use a letter like *a, b, x, y,* to stand for the number mentioned at that point, in the same way as in language "it" stands for a noun previously mentioned. The relation expressed by the remaining words will nearly always be one or more of addition, multiplication, and other well known operations of arithmetic.

In Boolean algebra, the things mentioned are either classes, or are converted by paraphrasing into classes. Since there is no standard notation for classes, we ordinarily have to assign letters to them, both when we know exactly what the class is, and also when we don't know what the class is. The relation expressed by the remaining words can usually be identified with little trouble as a combination of AND, OR, NOT, and other operations and relations of Boolean algebra.

4. Class Equality

Let us proceed with some more examples of Boolean algebra expressed in ordinary language:

EXPRESSION 2:

Employees with continuous service of at least five years are the members of the Great Lakes Pension Plan.

Translation:

(Employees with continuous service of at least *f*ive years) ARE THE (employees who are members of the Great Lakes Pension Plan).

f are the *g*.

f EQUALS *g*.

$f = g$.

Paraphrases:

Employees with continuous service of at least five years *constitute* the Great Lakes Company Pension Plan members.

Employees with continuous service of at least five years *are identical with* employees who are members of the Great Lakes Pension Plan.

Employees with continuous service of at least five years *are the same as* the employees who are in the Great Lakes Pension Plan.

This example illustrates two points of importance. First, so long as the members of each of the two classes are the same, EQUALS is true, no matter what the verbal description of the two classes may be; here the descriptions are very different, but the contents are the same. Second, the position of the word THE deserves notice. In "*A*'s are the *B*'s", which is the form of expression 2, the relation is equality. In "The *A*'s are *B*'s," which is the form of the second paraphrase under Expression 1, the relation is inclusion. The change in position of the word THE has completely altered the relation.

5. Class Inequality

EXPRESSION 3:

It is not true that employees with continuous service of at least five years are the same as the members of the Great Lakes Pension Plan.

Translation:

IT IS NOT TRUE THAT (employees with continuous service of at least five years) ARE THE SAME AS (the members of the Great Lakes Pension Plan).

It is not true that f is the same as g.

f IS NOT EQUAL TO g.

$f \neq g$.

Paraphrases:

Employees with continuous service of at least five years *are not identical with* members of the Great Lakes Pension Plan.

There is *a difference between* employees with continuous service of at least five years *and* members of the Great Lakes Pension Plan.

Possibly someone would suggest that another paraphrase might be: "Employees with continuous service of at least five years are not members of the Great Lakes Pension Plan." This is an ambiguous statement, for it is impossible to tell from it, grammatically correct though it may be, which of two meanings is meant: (*a*) NO (employees with continuous service of at least five years) ARE (members of the Great Lakes Pension Plan); or (*b*) (employees with continuous service of at least five years) ARE NOT IDENTICAL WITH (members of the Great Lakes Pension Plan).

Expression 3 is the contradiction of expression 2. But as in any mathematics, we may contemplate both $a = b$ and $a \neq b$ in the process of solving a problem.

A third point is illustrated by this expression and its translation. NOT here appears in a new role, referring to the relation between f and

g, instead of its old role, referring to classes, as in "Great Lakes Company employees who are NOT members of the Pension Plan." To escape confusion, it is necessary to distinguish between these two roles.

6. The Null Class

EXPRESSION 4:

There are no employees with $500 or more special retirement income annually.

Translation

THERE ARE NO (employees with $500 or more special retirement income annually).

There are no *s*.

s EQUALS THE NULL CLASS.

$s = 0$.

Paraphrases:

No employee has $500 or more of special retirement income annually.

Nobody has $500 or more of special retirement income annually.

It is *impossible* for any employee to have $500 or more of special retirement income annually.

It is *contrary to the rules* for any employee to have $500 or more of special retirement income annually.

If any employee should obtain $500 or more of special retirement income annually, he would *automatically cease* to have $500 or more of special retirement income annually.

The *number* of employees having $500 or more of special retirement income annually is *zero*.

There are *no instances* of employees with $500 or more of special retirement income annually.

A significant paraphrase in this case is "contrary to the rules"; most cases of impossibility are only cases of "contrary to the rules" and therefore depend on the rules. If the class of all things being discussed were Great Lakes Company employees as actually existing under the rules of reality, instead of Great Lakes Company employees as governed by the rules of the booklet, it might be quite possible for employees to have over $500 special retirement income annually.

7. Classes that are Not Empty

EXPRESSION 5:

There are employees receiving additional retirement income.

Translation:
THERE ARE (employees receiving *a*dditional retirement income).
There are *d*.
d IS NOT EQUAL TO THE NULL CLASS.
$d \neq 0$.

Paraphrases:
Some employees receive additional retirement income.
It is *in accordance with the rules* for an employee to received additional retirement income.
It is *possible* for an employee to receive additional retirement income.
There are *some instances* of employees receiving additional retirement income.
The *number* of employees receiving additional retirement income is *not zero*.

The second and third paraphrases of this expression explicitly call attention to the point that we are discussing booklet rules instead of reality. In the first, fourth, and fifth paraphrases this point is implicit, because in every accurate discussion the class of all things being discussed is always kept in mind. At some period in reality there might accidentally be no employees receiving additional retirement income; at the same time it would be quite in accordance with the rules for employees to receive additional retirement income, and the booklet would have to contain provisions with that in view.

8. AND; Recognizing of Classes in More than One Way

EXPRESSION 6:
male employees with at least five years of continuous service receiving additional retirement income

Translation a:
(male employees with at least five years of continuous service receiving additional retirement income)
e

Translation b:
employees who are (*m*ale employees) AND (employees having at least *f*ive years of continuous service) AND (employees receiving *a*dditional retirement income)
m and *f* and *d*
$m \cdot f \cdot d$
mfd

Paraphrases:

employees that are male, have at least five years of continuous service, *and* receive additional retirement income (note that the two commas as well as the "and" are also an important part of the language, signaling the relation)

employees who *meet all of the following conditions*: (1) are male; (2) have at least five years of continuous service; (3) receive additional retirement income.

Of course, $e = mfd$, since the contents of e are the same as the contents of mfd. The difference in the translations arises from the fact that the person applying Boolean algebra has some freedom in selecting the classes to be labeled and discussed. Which translation is the desirable one depends on the aim of the discussion.

Translation *b* also illustrates which one of the several meanings of the word AND is the meaning of the operator dot (·) in Boolean algebra, namely the AND which is equivalent to a restrictive relative clause, the AND which further specializes and delimits a kind of things.

9. The Negative of a Class

EXPRESSION 7:

nonmembers of the Great Lakes Pension Plan

Translation:

NON-(members of the Great Lakes Pension Plan)

NOT g

g'

EXPRESSION 7*a*:

all Great Lakes Company employees who are nonmembers of the Great Lakes Pension Plan

Translation:

employees who are (employees of the Great Lakes Company) BUT NOT (members of the Great Lakes Pension Plan)

1 and not g

$1 \cdot g'$

What difference is there between these two expressions? One answer might be that 7*a* seems precise and clear while 7 seems ambiguous. In cases where the persons engaged in discussion have agreed upon the universe class, the class of all the things being discussed, 7 and 7*a* are exactly equivalent: $g' = 1 \cdot g'$ in Boolean algebra, just as $(-8) = 1$ TIMES -8 in elementary algebra; and expression 7 is precise and clear. This is the case in the present situation, since we have agreed that we are

discussing all Great Lakes Company employees. But in cases where the persons discussing have not settled on a universe class, have not selected the same territory within which to discuss, NOT is ambiguous, denoting different things to different persons, expression 7 is vague, and the application of Boolean algebra falters for lack of obedience to its rules.

Paraphrases of expression 7 or 7a:

all Great Lakes Company employees *except* those in the Great Lakes Pension Plan

employees *not* members of the Great Lakes Pension Plan

employees *other than* members of the Great Lakes Pension Plan

10. Exception

EXPRESSION 8:

employees who fail to enter the Pension Plan when eligible, except employees who when eligible are not actively at work

Translation:

(employees who fail to enter the Pension Plan when eligible) EXCEPT (employees who when eligible are not actively at work [are sick])

t except k

t and not k

$t \cdot k'$

Paraphrases:

employees who fail to enter the Pension Plan when eligible, *but not* employees who, when eligible, are not actively at work

employees who fail to enter the Pension Plan when eligible, *other than* employees who, when eligible, are not actively at work

employees who fail to enter the Pension Plan when eligible *unless* such employees, when eligible, are not actively at work

employees who fail to enter the Pension Plan when eligible, *exclusive of* employees who, when eligible, are not actively at work

11. OR (AND/OR, OR ELSE)

EXPRESSION 9:

all employees entitled to annual retirement income or special retirement income

Translation:

(employees entitled to annual retirement income) OR (employees entitled to special retirement income)

a or p or both

$a \vee p$

Paraphrases:

employees entitled to annual retirement income *or* special retirement income *or both*

employees entitled to annual retirement income *and/or* special retirement income

employees who meet *one or both of the following conditions* (1) are entitled to annual retirement income; (2) are entitled to special retirement income

The meaning of the operator OR (v) in Boolean algebra is the meaning of OR which does not exclude the situation of both, several, or all of the alternatives mentioned. The "exclusive" OR is illustrated just below:

EXPRESSION 10:

employees entitled to annual retirement income or special retirement income but not entitled to both

Translation:

(employees entitled to *a*nnual retirement income) OR (employees entitled to *s*pecial retirement income) BUT NOT (employees entitled to BOTH *a*nnual AND *s*pecial retirement income)

a or p but not both a and p

$(a \vee p) \cdot (ap)'$

Paraphrases:

employees entitled to annual retirement income *or else* to special retirement income

either employees entitled to annual retirement income *or* employees entitled to special retirement income

employees entitled to *one or the other* of annual retirement income and special retirement income, *excluding* those entitled to *both*

The word OR is frequently used in the meaning "or else" and frequently used in the meaning "and/or". This ambiguous usage is one of the most troublesome of the ambiguous features of ordinary language. The ambiguity vanishes in Boolean algebra since the different meanings are differently symbolized. But the person translating from ordinary words into Boolean algebra must be able to decide correctly from the context which one of AND/OR or OR ELSE is really meant.

12. The Null Class and the Universe Class

EXPRESSION 11:

No employees are both male and female.

Translation:
(No employees) ARE BOTH (*m*ale) AND NOT (*m*ale).
0 are both m and not m.
$0 = m \cdot m'$

Paraphrases:
There are no employees *both* male *and not* male.
Employees *each of whom is both* male *and not* male *number zero.*

EXPRESSION 12:
All employees are either male or female.

Translation:
(All employees) ARE EITHER (*m*ale) OR NOT (*m*ale).
1 is either m or not m.
$1 = (m \vee m')(m \cdot m')'$

Paraphrases:
Any employee *is either* male *or else* not male.

13. ONLY .... MAY BE ....

EXPRESSION 13:
Only employees who are clerks may obtain special retirement income.

Translation:
ONLY (employees who are *c*lerks) MAY BE (employees who obtain *s*pecial retirement income).
Only c may be p.
All p are c.
$p \subset c$.

Paraphrases:
Only clerks *are eligible for* special retirement income.
Among employees who are clerks *will be found all* employees that obtain special retirement income.
Employees who are clerks *include all* employees that obtain special retirement income.

It is worth noting that this expression is simply the converse of "all .... are .... ."

14. A Final Example

EXPRESSION 14:
Employees leaving the service of the Great Lakes Company are not allowed a withdrawal value for retirement income which might have been

secured had they continued in service, except in the cases of female employees with at least one year's continuous service who are leaving to be married, giving advance notice to that effect.

Translation:

(Employees leaving [*r*esigning from] the service of the Company) ARE {NOT (employees allowed a withdrawal value for retirement income *q*ualified for)}; except: employees who are NOT (*m*ale) AND have (at least one year of continuous service) AND are (leaving to be married [*w*edded]) AND (give advance *n*otice of leaving) ARE (employees allowed a withdrawal value for retirement income *q*ualified for).

All r are (not q), except r and not m and o and w and n are q.

$rm'own \subset q$; $r(m'own)' \subset q'$.

Several points are brought out by this example. First, in cases where letters for symbols for classes have been used already, the initial letter of a synonym for an important word in the description of a class may furnish a satisfactory symbol. Second, in many phrases where the concept of "employees" seems to be missing, it can be made to appear, and thus establish a class, for class is the foundation element of the Boolean algebra of classes as number is the foundation element of arithmetic. Third, the word "except" in this expression acts more to describe the relation between two sentences than to describe a relation between two classes.

15. A Summary of Translations

We can sum up the translations into Boolean algebra of ordinary expressions of English in Table 4-2.

TABLE 4-2

Ordinary Words	Standard Boolean Words*	Standard Boolean Algebra
1. All a's are b's	a lies in b	$a \subset b$
2. All a's are not b's	AMBIGUOUS—see No. 3 and No. 17	.....
3. All a's are non-b's	a lies in not-b	$a \subset b'$
4. An a is a b	a lies in b	$a \subset b$
5. a's and b's	AMBIGUOUS—see No. 7, No. 22, No. 25	.....
6. Any a is a b	a lies in b	$a \subset b$
7. What is both an a and a b	a and b	$a \cdot b$
8. a's except b's	a and not-b	$a \cdot b'$

TABLE 4-2 (*Continued*)

Ordinary Words	Standard Boolean Words*	Standard Boolean Algebra
9. everything is an a	a equals the universe class	$a = 1$
10. a's excluding b's	a and not-b	$a \cdot b'$
11. if it is an a, then it is a b	a lies in b	$a \subset b$
12. it is an a if and only if it is a b	a equals b	$a = b$
13. a's are included in b's	a lies in b	$a \subset b$
14. no a's are b's	a and b is empty	$a \cdot b = 0$
15. neither a's nor b's	not-(a or b)	$(a \vee b)'$
16. not both a's and b's	not-(a and b)	$(a \cdot b)'$
17. not all a's are b's	a and non-b is not empty	$a \cdot b' \neq 0$
18. none of the a's are b's	a and b is empty	$a \cdot b = 0$
19. a's or else b's	a and not-b, or b and not-a	$a \cdot b' \vee b \cdot a'$
20. a's or b's but not both	a or b, and not (a and b)	$(a \vee b) \cdot (a \cdot b)'$, which reduces to $a \cdot b' \vee b \cdot a'$
21. a's or b's	AMBIGUOUS—see No. 19 and No. 22	.....
22. a's or b's or both	a and/or b, a or b	$a \vee b$
23. it is an a only if it is a b	a lies in b	$a \subset b$
24. some a's are b's	a and b is not empty	$a \cdot b \neq 0$
25. a's that are b's	a and b	$a \cdot b$
26. b's that are a's	b and a	$b \cdot a$, which is the same as $a \cdot b$
27. there is no a	a is empty	$a = 0$
28. there are no a's	a is empty	$a = 0$
29. there is an a	a is not empty	$a \neq 0$
30. there are some a's	a is not empty	$a \neq 0$
31. there exists an a	a is not empty	$a \neq 0$

**Note*: The Boolean words "or" and "and" are not ambiguous.

Chapter 5

BOOLEAN ALGEBRA: CALCULATING

1. A Summary of the Rules for Symbolic Calculating with Boolean Algebra

Now, how can we calculate in Boolean algebra?

There are about three dozen rules in Boolean algebra which are sufficient to solve nearly all problems. These are the rules, expressed in both symbols and words:

1. $a \vee b = b \vee a$	a or b is the same as b or a
2. $(a \vee b) \vee c = a \vee (b \vee c)$	(a or b) or c is the same as a or (b or c)
3. $a \cdot b = b \cdot a$	a and b is the same as b and a
4. $(a \cdot b) \cdot c = a \cdot (b \cdot c)$	(a and b) and c is the same as a and (b and c)

These four rules are just like the rules in ordinary algebra for PLUS and TIMES. The first and third are called the commutative laws; the other two are called the associative laws.

5. $a(b \vee c) = ab \vee ac$	a and (b or c) is the same as a and b or a and c
6. $a \vee bc = (a \vee b)(a \vee c)$	a or (b and c) is the same as (a or b) and (a or c)

These are the two distributive laws in Boolean algebra. In the ordinary algebra of numbers there is only one distributive law: $a \cdot (b + c) = a \cdot b + a \cdot c$

7. $a \vee a = a$	a or a is the same as a
8. $a \cdot a = a$	a and a is the same as a

The first rule is true in ordinary algebra only for the number 0. The second rule is true only for the numbers 0 and 1. In Boolean algebra, however, these rules are true for all classes without exception.

9. $a \vee (a \cdot b) = a$	What is a or (a and b) is the same as a
10. $a (a \vee b) = a$	What is a and (a or b) is the same as a

These two rules are called the Law of Absorption, since the extra term or coefficient in the expression is "absorbed".

We now come to six rules about the universe class and the null class:

11. $a \vee 0 = a$	What is a or nothing is the same as a
12. $a \cdot 1 = a$	What is both a and everything is the same as a
13. $a \vee 1 = 1$	What is a or everything is the same as everything
14. $a \cdot 0 = 0$	What is both a and nothing is the same as nothing
15. $a \vee a' = 1$	Everything is a or not-a. (Stated in the reverse order because of English idiom.)
16. $a \cdot a' = 0$	Nothing is both a and not-a. (Stated in reverse order.)

Some rules about negation:

17. $a \vee b = (a' \cdot b')'$	a or b is the same as not (not-a and not-b)
18. $(a \vee b)' = a' \cdot b'$	What is neither a nor b is the same as what is both not-a and not-b
19. $a \cdot b = (a' \vee b')'$	a and b is the same as not (not-a or not-b)
20. $(a \cdot b)' = a' \vee b'$	What is not both a and b is the same as what is not-a or not-b
21. $ab \vee ab' = a$	What is a and b, or a and not-b, is the same as a
22. $(a \vee b)(a \vee b') = a$	What is both (a or b) and (a or not-b) is the same as a
23. $(a')' = a$	What is not not-a is the same as a
24. $0' = 1$	What is not-nothing is everything
25. $1' = 0$	What is not everything (idiomatically, not anything) is nothing

Rules about the relation LIES IN, inclusion between classes:

26. $a \subset b$ is equivalent to $a \cdot b' = 0$ or $ab = a$ or $a' \vee b = 1$ or $a \vee b = b$ or $b' \subset a'$. To express this in words, we say that the following statements are all interchangeable:
   (i) a lies in b.
   (ii) Nothing is both a and not-b.
   (iii) What is both a and b is the same as a.
   (iv) Everything is not-a or b.
   (v) What is a or b is the same as b.
   (vi) Not-b lies in not-a.

An expansion of previous rules to apply to more classes:

27. $1 = (a \text{ v } a')(b \text{ v } b')(c \text{ v } c') \ldots .$
    Everything is either a or not-a, and b or not-b, and c or not-c . . . .
28. $(a \text{ v } b \text{ v } c \text{ v } \ldots .)' = a'b'c' \ldots .$
    What is not (a or b or c or . . . .) is the same as what is not-a and not-b and not-c . . . . .

Rules about the relation of equality between classes:

29. $a = b$ is equivalent to $ab' \text{ v } a'b = 0$ or $a \text{ v } b = ab$ or $ab \text{ v } a'b' = 1$ or both $a \subset b$ and $b \subset a$.
    The following statements are all interchangeable:
    (i) The class a equals the class b.
    (ii) Nothing is a and not-b, or b and not-a.
    (iii) Everything is both a and b or both not-a and not-b.
    (iv) a lies in b and b lies in a.

Some rules for handling equations, functions, and ranges of values:

30. $x \text{ v } y = 0$ is equivalent to $x = 0$ and $y = 0$.
    The following statements are interchangeable:
    (i) Nothing is x or y.
    (ii) Nothing is x and nothing is y.
31. $x \text{ v } y = 1$ and $xy = 0$ is equivalent to $x = y'$ or $y = x'$.
    The following statements are interchangeable:
    (i) Everything is x or y, and nothing is both x and y.
    (ii) x is not-y.
    (iii) y is not-x.
32. Any expression may be put into the form $Ax \text{ v } Bx'$, the normal form, $f(x)$, of a function of one variable x. Then:
    $f(1) = A$
    $f(A) = A \text{ v } B$
    $f(x) = f(1) \cdot x \text{ v } f(0) \cdot x'$
    $f(B) = AB$
    $f(0) = B$
    $AB \subset f(x) \subset A \text{ v } B$ for every x.
33. The equation $Ax \text{ v } Bx' = 0$ is equivalent to $B \subset x \subset A'$, or $x = Bu' \text{ v } A'u$ for any u.
34. For any a, $0 \subset a \subset 1$.
35. For any a and b, it is not necessarily true that either $a \subset b$ or $b \subset a$.

2. An Example of Calculation

What is an example of the use of these rules? Although many examples of calculation employing these rules will be given in later chapters, let us take John Venn's problem stated p. 22 and solve it here, showing how the rules for calculation may be applied.

Problem: A certain club has the following rules: (1) The financial committee shall be chosen from among the general committee. (2) No one shall be a member of both the general and library committee unless he is also on the financial committee. (3) No member of the library committee shall be on the financial committee. Simplify these rules.

Solution: Let f = members on the financial committee, g = members on the general committee, and b = members on the library committee. (*Note*: Usually a good choice for a symbol for a class is the first letter of the chief descriptive word. But the letters l and O in typewriting are not usually distinguishable from the numbers 1 and 0, and so are avoided.)

The given conditions may be translated into:

(1) All f are g.

$f \subset g$

$fg' = 0$ By Rule 26

(2) All that are both g and b are f.

$gb \subset f$

$gbf' = 0$ By Rule 26

(3) There are no members both f and g.

$bf = 0$

$fg' \vee gbf' \vee bf = 0$	To combine given conditions, put each in a form equal to the null class and connect with OR. (Rule 30)
$fg' \vee bf'g \vee bf = 0$	Rearranging each term into alphabetic order for more systematic processing.
$fg' \vee bf'g \vee bf(g \vee g') = 0$	Using the rule $a = a(c \vee c')$ (Rule 21)
$fg' \vee bf'g \vee bfg \vee bfg' = 0$	Using the rule $a(c \vee c') = ac \vee ac'$ (Rule 5)
$(fg' \vee bfg') \vee (bf'g \vee bfg) = 0$	Regrouping
$fg' \vee bg(f \vee f') = 0$	Using $a \vee ac = a$, and $ab \vee ac = a(b \vee c)$. (Rules 9, 5)
$fg' \vee bg = 0$	Using $a \vee a' = 1$, and $a \cdot 1 = a$ (Rules 15, 12)

$fg' = 0$, $bg = 0$	If $x \vee y = 0$, then $x = 0$, and $y = 0$. (Rule 30)
$f \subset g$	If $ab' = 0$, then $a \subset b$. (Rule 26)

Translating the two simplified statements:

Answer: The rules may be simplified as follows: (1) The financial committee shall be chosen from among the general committee. (2) No member of the general committee shall be on the library committee.

3. Why are the Rules of Calculation in Boolean Algebra True?

To answer this question adequately, we need first to distinguish between truth based on observations and truth based on logical reasoning, between "facts" and "valid arguments", between observational truth and demonstrated truths.

From the point of view of a careful observer of the world, these rules are true because they sum up observed properties and relations of classes with no observed exceptions. We can say for example that scientists have studied classes and their relations, and have summed up their observations in these empirical rules. For example, in the next section of this chapter, we ourselves will take a look at graphic representations of dots on a sheet of paper, and we ourselves will be able to make enough observations to satisfy ourselves that many of these rules are very likely to be true.

From the point of view of a mathematician or logician, however, these rules are true for a different reason: namely, that they all flow logically from a few fundamental starting concepts and assumptions. If these "undefined" concepts" and "unproved assumptions" are granted, then all that follows must be logically true. All the rules for calculation given above are either included in the assumptions or can be proved by the methods of logic from these assumptions.

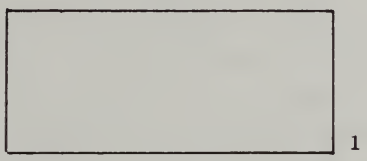
Furthermore, Boolean algebra can be proved to be consistent, free from contradictions, in the sense that it can be proved that no statement in Boolean algebra and its denial can both be proved.

It is worth noting that, for elementary algebra (the algebra of ordinary numbers) it has been discovered that the absence of contradiction cannot be proved. This is a recent and disconcerting discovery. So one might claim facetiously that Boolean algebra is a better algebra than elementary algebra.

4. A Summary of the Rules for Graphic Calculating with Boolean Algebra

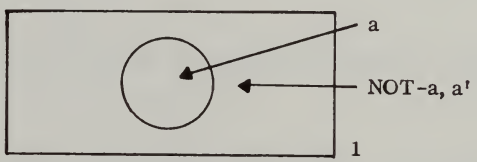
How can classes, and their properties and relations, be expressed graphically?

We represent the universe class, the class of all the things being discussed, by a rectangle:

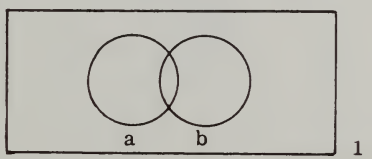


The classes mentioned may overlap, each with all the others. So we draw them as closed areas inside the rectangle, drawing them so that each class overlaps every combination of the others. NOT-*a* is the area in the rectangle outside of *a*.

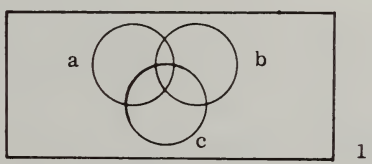
For one class *a*:



For two classes *a* and *b*:

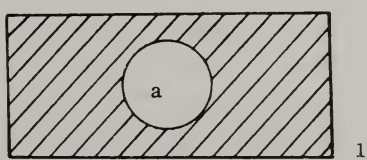


For three classes *a*, *b*, and *c*:



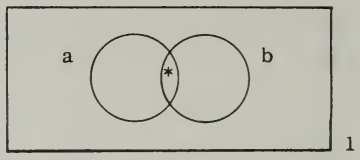
The null class has no specific location. But a class which is determined to be empty is represented by an area shaded through:

NOT-*a* is empty: $a' = 0$:



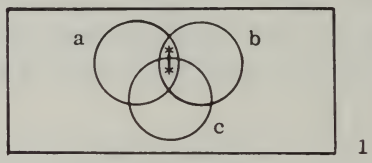
A class which is known or proved to be not empty is marked by placing an asterisk in the area:

Some a's are b's:
$ab \neq 0$:



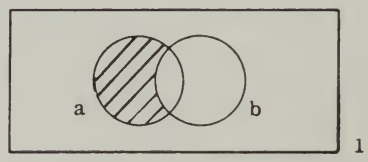
If it is not clear just where the non-emptiness is, the asterisk is connected by a line with other asterisks, to suggest the ambiguity of the location of the nonemptiness:

Some a's are b's:
$ab \neq 0$:

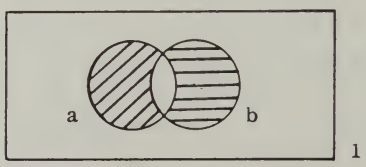


Here are some examples of the relations of classes expressed graphically:

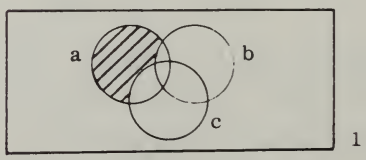
All a is b.
$a \subset b$
$ab' = 0$



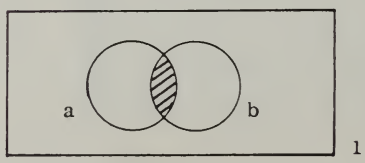
a is the same as b.
$a = b$
$ab' \vee a'b = 0$



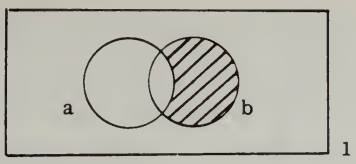
All a is b or c.
$a \subset b \vee c$
$a(b \vee c)' = 0$



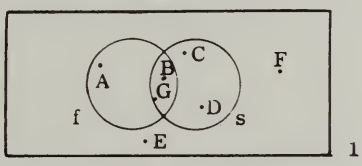
No a is b.
$ab = 0$



Only a may be b.
$b \subset a$
$ba' = 0$



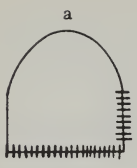
If we desire to represent the individual things contained in classes, we may use dots. For example, suppose we are discussing all the employees in the Inspection Department of the Great Lakes Company on January 1, 1938 (who happen to be Adams, Brown, Cohen, Davis, Edwards, Fields, and Gray), and those employees who have had continuous service of at least *f*ive years (who happen to be Adams, Brown, and Gray) and those employees who are *s*teamboat inspectors (who happen to be Brown, Cohen, Davis and Gray). This situation may be graphically diagrammed as follows:



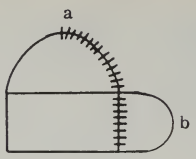
In this case, a tremendous variety of diagrams is possible, all equally good. For example, any other boundary that enclosed A, B, and G would equally well represent f, for, by the definition, if the things in the class a are the same as the things in the class b, then $a = b$. Now appears the reason why the null class has no specific location: it could be placed anywhere and made of any size so long as it contained none of the seven points, A, B, C, D, E, F, and G.

It is unnecessary that the boundary of a class should be a single closed curve. However, it is so much easier to visualize the relations of classes when each is a single area that much effort has gone into constructing suitable diagrams for 4, 5, 6, ... , classes where each is bounded by a single curve and where the boundaries, taken together, overlap to give 16, 32, 64, ... , subdivisions.

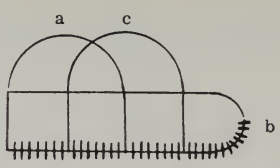
The simplest solution seems to be expressed in the sequence of diagrams on page 50 (each diagram is to be imagined as placed inside of a rectangular universe class), where the rule for proceeding from diagram n to diagram $n + 1$ might be stated: Select alternate lines in the rectangular portion of diagram n and mark them. Follow these lines into the curved portion and mark them there also. Using the marking as a sketch of the location of the required additional class, redraw the diagram, including the new class. Thus, in proceeding from diagram 4 to



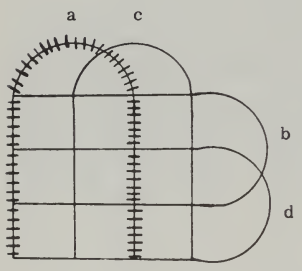
1.



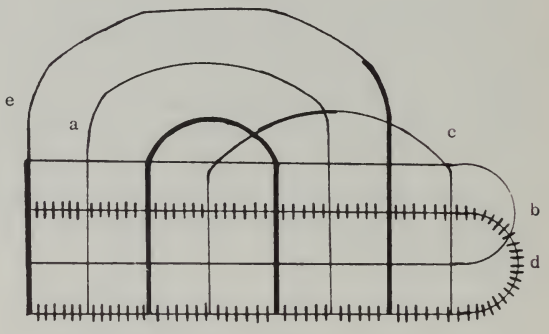
2.



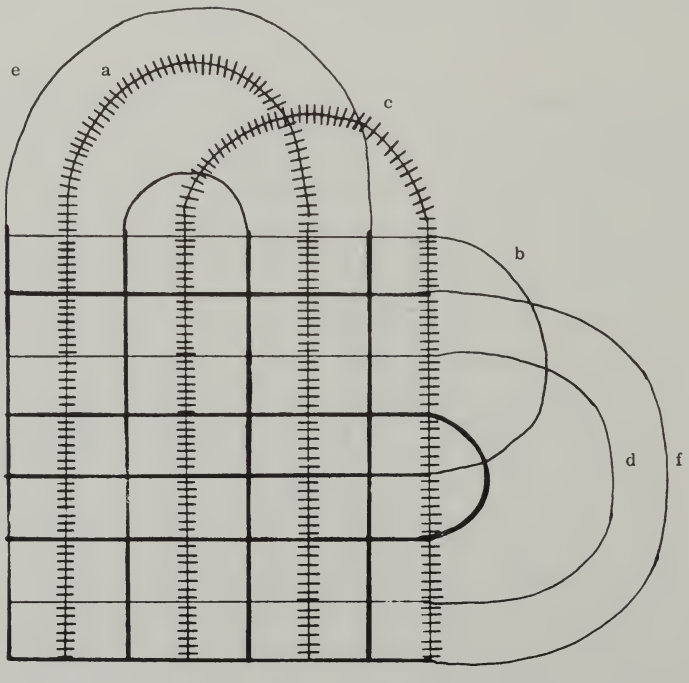
3.



4.



5.

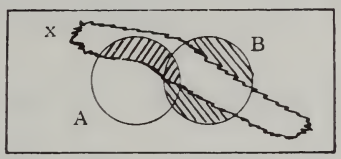


6.

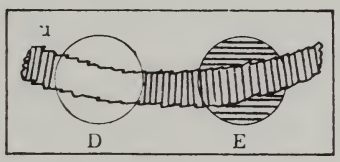
diagram 5, the segment of the boundary of the class a, marked with crosshatching, sketches out the location and horseshoe shape of class e.

Boolean algebra functions, such as $f(x) = Ax \vee Bx'$, and equations, such as $Dy \vee Ey' = 0$ (solution: $y = Eu' \vee D'u$ for any u); may be represented graphically as follows:

For x as shown, f is the cross-hatched area.



For some u as shown, the solution y is the cross-hatched area.

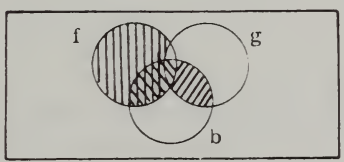


5. An Example of Graphic Calculation

What is an example of how these rules for graphic calculation can be applied? Let us take again the same problem due to John Venn about the three committees (the statement of the problem is on page 45):

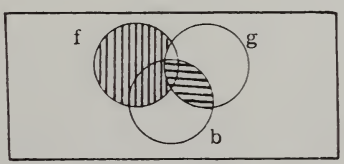
The three different conditions given may be expressed each by different shading:

(1) $fg' = 0$
(2) $gbf' = 0$
(3) $bf = 0$



Looking at the diagram we see that we can blank out exactly the same areas by two shadings instead of three:

(1) $fg' = 0$
(2) $bg = 0$



These two shadings are the simplified rules.

Chapter 6

BOOLEAN ALGEBRA: MATHEMATICAL DEFINITION

1. The Mathematical Definition of a Boolean Algebra

So far we have referred to three interpretations of Boolean algebra:

(1) the *algebra of classes,* which has been explained at length above;
(2) the *algebra of on-off circuit elements,* which has been mentioned only once but which will be explained and illustrated at length in later chapters;
(3) the *algebra of sets of points or regions,* which we made use of to provide graphic calculation in Boolean algebra.

There are a number of other interpretations, all of them interesting and some of them important. But first let us ask, When is something or other a Boolean algebra? What is the definition of Boolean algebra?

A mathematical system is a Boolean algebra when it meets (fulfills) a certain set of requirements stated below (or other sets of requirements interchangeable with that one). A set of requirements which defines a mathematical system is called a set of *postulates.*

Postulates are demands. The postulates of Boolean algebra are those demands which, when granted, permit all the rest of Boolean algebra to be logically deduced. In other words, postulates in a system are the fundamental statements or axioms which lead by logical steps to every other statement in the system. The postulates consist of concepts and propositions which are accepted as given. Other concepts are then set up by definition, and other propositions established by reasoning.

There are several ways of selecting a set of postulates for Boolean algebra. One of the possible sets is stated below. The postulates of Boolean algebra are useful in checking any proposed mathematical system to see if it is actually behaving like Boolean algebra.

In essence, checking a mathematical system against a set of postulates is just like comparing sample cases against general statements to see if the general statement is false for any one case or true for all cases. Any situation where a number of individual cases are studied and compared with rules to govern them is substantially a situation in

the "mathematical theory of postulates", the theory of fundamental statements.

2. A Set of Postulates

The following is a set of postulates defining Boolean algebra, given by E. V. Huntington in 1904. Any mathematical system which is truthfully described by these postulates is a Boolean algebra.

Concepts accepted as given:

$a, b, c, \ldots$ Elements; which may be thought of as items in a list K (possibly an infinite list)

$\oplus$ (read "pseudoplus") An operation; which may be thought of as a rule by means of which any two items in the list, such as a and b, determine another item or items (maybe in this list, maybe not), collectively called $a \oplus b$. The rule in its essence is a two-way table showing for every value of a along the side and for every value of b along the top what item or items $a \oplus b$ is or are. Although the plus sign (+) is used as part of $\oplus$, the operation need have no resemblance to addition.

Propositions accepted as given:

(1) $a \oplus a = a$.
(2) $a \oplus b = b \oplus a$.
(3) $(a \oplus b) \oplus c = a \oplus (b \oplus c)$.
(4) There exists at least one element z such that $a \oplus z = a$ for every a.
(5) There exists at least one element u such that $u \oplus a = u$ for every a.
(6) $a \oplus b$ determines an element in K.
(7) There exists at least one element a' for every a such that: if $x \oplus a = a$, and if $x \oplus a' = a'$, then $x = z$ (where z is an element such that $b \oplus z = b$ for every b) and $a \oplus a' = u$ (where u is an element such that $u \oplus c = u$ for every c).
(8) If $a \oplus b' \neq b'$, there exists at least one element x such that $a \oplus x = a$ and $b \oplus x = b$ and yet for some c, $c \oplus x \neq c$.

Definitions:

(1) $a \subset b$ is equivalent to $a \oplus b = b$.
(2) $a \odot b = (a' \oplus b')'$.
(3) u, which turns out to be unique, since it can be quite easily proved from the postulates that there is only one u satisfying the postulates.
(4) z, which turns out be be unique, since it can likewise be easily proved that there is only one z satisfying the postulates.

In other words, if we come across a collection or list of mathematical elements (which of course can be called $a, b, c \ldots .$) and an operation (which of course can be called pseudoplus, $\oplus$, which associates two of them and gives a third (which may or may not be the same), then we can find out, by testing the behavior of this mathematical system (i.e., collection and operation) against each of the foregoing eight statements, whether or not this system is a Boolean algebra.

Here, then, is a definition of a Boolean algebra. Not only does it give a set of criteria for testing, but in addition every rule and theorem of Boolean algebra can be proved rigorously from these postulates. Also, it happens that the postulates in this set are independent: not one of them can be proved from the others in any combination, as can be shown by eight cleverly chosen examples of mathematical systems, one corresponding to each postulate. Finally, it should be remarked that there are of course, other sets of postulates for Boolean algebra besides this one.

3. Interpretations of Boolean Algebra

The postulates for Boolean algebra describe at least half a dozen different mathematical systems. In other words, the concepts and rules expressed in the postulates may be interpreted to tell the truth about at least half a dozen different features of reality.

Interpretation 1: *CLASSES*: In this interpretation, the meanings assigned to the symbols appearing in the postulates are:

$a, b, c, \ldots .$	classes
$a \oplus b$	a v b, a OR b OR BOTH
$a \odot b$	$a \cdot b$, a AND b
a'	a', NOT-a
$a \Subset b$	$a \Subset b$, a IS CONTAINED IN b
u	1, THE UNIVERSE CLASS, EVERYTHING
z	0, THE NULL CLASS, NOTHING

This is the interpretation which we have used to introduce and make clear the meaning of Boolean algebra.

For a concrete example, suppose we are discussing all the employees in the Inspection Department of the Great Lakes Company (who happen to be Adams, Brown, Cohen, Davis, Edwards, Fields, and Gray) and those who have had continuous service of at least five years (who happen to be Adams, Brown and Gray). We have immediately begun to deal with a Boolean algebra, where

1 = All employees in the Inspection Department (that is, the class containing as members A, B, C, D, E, F, and G).

f = Those employees in this department with continuous service of at least *five* years (that is, the class containing as members *A*, *B*, and *G*).

f' = Those employees in this department who do not have continuous service of at least five years (that is the class containing as members *C*, *D*, *E*, and *F*).

0 = The null class, the class of no employees.

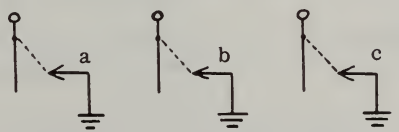
On the list *K* of the elements of the mathematical system we shall have just four items, 1, f, f' and 0, and no more. For the table of the operation $\oplus$ we shall have:

$\oplus$	1	f	f'	0
1	1	1	1	1
f	1	f	1	f
f'	1	1	f'	f'
0	1	f	f'	0

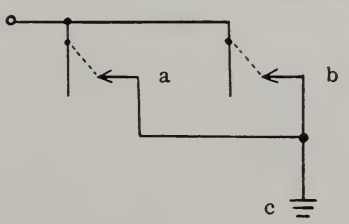
And if we verify each one of the postulates using this sample mathematical system, we shall find that the postulates truthfully describe this system. Consequently, this system is proved to be a Boolean algebra.

Interpretation 2: *ON-OFF CIRCUIT ELEMENTS*: In this interpretation, the meanings assigned to the symbols in the postulates are:

a, b, c, ... on-off circuit-elements, for example, switch contacts, which may be closed or not closed, or wires which may carry current or not carry current, etc. In the diagram, the small circle o marks the "source" of current and ⏚ marks the "sink" of current, or ground, and the mark ↖ indicates a moving switch contact.

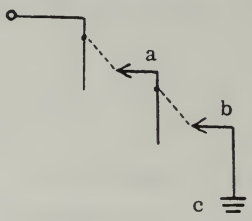


$a \oplus b$ *a* and *b in parallel:*



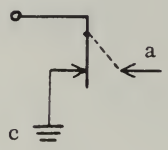
$c = a \vee b$; that is, current is in *c* if *a* is energized OR *b* is energized OR BOTH

$a \odot b$ *a* and *b in series*:



$c = a \cdot b$; current is in *c* if *a* is closed AND *b* is closed.

a'

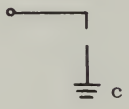


$c = a'$: current is in *c* when *a* is not closed.

u

Current is in *c* always: $c = 1$

z



Current is in *c* never: $c = 0$

Interpretation 3: *REGIONS*

$a, b, c, \ldots$	Regions in a plane
$a \oplus b$	The region containing *a* and containing *b* but counting any common region only once; the *join* of *a* and *b*
$a \odot b$	The region common to *a* and *b*; the *union* of *a* and *b*
a'	The region outside of *a*
$a \Subset b$	The region *a* LIES IN the region *b*
u	The whole plane, everywhere
z	Nowhere

Interpretation 4: *FACTORS OF CERTAIN NUMBERS*

$a, b, c, \ldots$	The factors of 30 (in other words, any of the numbers 1, 2, 3, 5, 6, 10, 15, 30)
$a \oplus b$	The Least Common Multiple of *a* and *b*
$a \odot b$	The Highest Common Factor of *a* and *b*
a'	30 divided by *a*
$a \Subset b$	*a* is a factor of *b*
u	30
z	1

Any number which contains no prime factor more than once may be used in place of 30.

Interpretation 5: ZERO and INFINITY:

$a, b, c, \ldots$ The numbers 0 ZERO and ∞ INFINITY

$a \oplus b$ a PLUS b, as defined by the table:

$0 + 0 = 0$ or

$0 + \infty = \infty$

$\infty + 0 = \infty$

$\infty + \infty = \infty$

	0	∞
0	0	∞
∞	∞	∞

$a \odot b$ a TIMES b, defined by the table:

	0	∞
0	0	0
∞	0	∞

a' THE OPPOSITE OF a, defined by

a	a'
0	∞
∞	0

u ∞

z 0

4. Interpretation of the Truth Values of Propositions

There is one more interpretation of Boolean algebra which is so important that it deserves to be given a lot of attention in its own right. This interpretation may be called the "algebra of truth values" or the "algebra of propositions". We shall explain a "truth value" and a "proposition" in a moment.

In this interpretation, the meanings assigned to the main symbols appearing in the postulates are these:

$a, b, c, \ldots$ variables each standing for the number 1 or 0; binary variables

$a \oplus b$ a v b, which is defined by the table:

a	b	a v b
0	0	0
0	1	1
1	0	1
1	1	1

$a \odot b$ — $a \cdot b$, which is defined by the table:

a	b	a, b
0	0	0
0	1	0
1	0	0
1	1	1

a' — a', which is defined by the table:

a	a'
0	1
1	0

u — 1, the number ONE

z — 0, the number ZERO

It is worth noting that the number of variables a, b, c, ... and so on can be large, just as in elementary algebra; but the number of different elements in this mathematical system is exactly two, 1 and 0; whereas in elementary algebra, the number of different elements is infinite, being the number of different numbers such as 2, 3 and 1/7, .1429, $\sqrt{10.2}$, and so on.

What is a proposition? and what is a truth value?

A *proposition* is a statement or an assertion which can be true or false, thought of or considered irrespective of the particular words or language used to express the statement. For example, "two and three are five", "deux et trois font cinq", "2 + 3 = 5", "zwei und drei fünfe sind", all express one and the same proposition; this proposition, of course, is true.

One proposition is the same as another proposition only when both propositions "say exactly the same thing". Ordinarily, we would expect the subject and the predicate of each proposition would have to have the same meaning or denotation, respectively. A proposition is often represented by a capital letter P, or Q, or R.

The *truth value* of a proposition P is "truth" if the proposition is true, and "falsity" if the proposition is false. The truth value may be represented by words or signs such as "yes, checkmark (✓) T, 1, true, correct, truth, right," etc. The truth value "falsity" may be represented by words or signs such as "no, not so, crossmark (×), F, 0, false, falsity, wrong, incorrect", etc.

Of all these representations, 1 and 0 are the best for purposes of ordinary calculating because of the possibilities of numerical computing, treating 1 and 0 as numbers; and we shall regularly use 1 and 0 for truth values henceforth.

The truth value of one proposition can be the same as the truth value of another proposition without any necessary relation whatever between the propositions.

It is convenient to express "the truth value of ...." as T(....), and to write $T(P) = p$ where P is a proposition; if Q is another proposition, we write $T(Q) = q$. In this way, when we do not know the truth value p of a proposition, we can still talk definitely about p, in the same way that when we do not know the actual value of the number of elements in some collection, we can still talk definitely about n.

The three common relations or connectives of statements in Boolean algebra are: OR, AND, NOT. If P and Q are statements, then the truth value of the statement P OR Q, depending on the truth value of P and Q, is shown in the following table:

P	Q	P OR Q
0	0	0
0	1	1
1	0	1
1	1	1

The truth value of the statement P AND Q is shown in the following table:

P	Q	P AND Q
0	0	0
0	1	0
1	0	0
1	1	1

The truth value of the statement NOT-P (the denial of P) is shown in the following table:

P	NOT-P
0	1
1	0

It can be easily shown that:

T (P OR Q OR BOTH) $= p \vee q$

T (P AND Q) $= p \cdot q$

T (NOT-P)	$= p'$
T (P OR ELSE Q)	$= pq' \vee p'q$
T (NEITHER P NOR Q)	$= p' \cdot q'$
T (NOT-P OR Q)	$= p' \vee q$
T (IF P, THEN Q)	$= p' \vee q$
T (P IF AND ONLY IF Q)	$= pq \vee p'q'$
T (P IS TRUE)	$= p$
T (P IS FALSE)	$= p'$
T (P IS FALSE OR Q IS TRUE)	$= p' \vee q$
T (ANY TRUE PROPOSITION)	$= 1$
T (ANY FALSE PROPOSITION)	$= 0$

Now, since 1 and 0 are numbers, we can ask this question: Are there some expressions of the ordinary elementary algebra of numbers which will give us $p \vee q$, $p \cdot q$, and p'? The answer is yes. A little experimenting shows:

p	q	p TIMES q $= pq$	p PLUS q MINUS pq $= p + q - pq = p \vee q$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

p	1 MINUS p $= 1 - p = p'$
0	1
1	0

Proceeding with this algebra of the numbers 1 and 0, it is easy to show that:

T (P OR ELSE Q)	$= p + q - 2pq$
T (NEITHER P NOR Q)	$= (1 - p)(1 - q) = 1 - p - q + pq$
T (IF P, THEN Q)	$= 1 - p + pq$
T (NOT-P OR Q)	$= 1 - p + pq$
T (P IF AND ONLY IF Q)	$= 1 - p - q + 2pq$
T (P IS FALSE OR Q IS TRUE)	$= 1 - p + pq$

This brings us to another fact, that the relations of truth values have truth values. So we can readily write down the following:

p	q	T $(p = q)$	T $(p \neq q)$	T $(p > q)$
0	0	1	0	0
0	1	0	1	0
1	0	0	1	1
1	1	1	0	0

p	T $(p = 1)$	T $(p = 0)$
0	0	1
1	1	0

Numerical operations that do not lead to numbers outside of 1 and 0 have application. For example:

$$\text{T } (P \text{ OR ELSE } Q) = (p - q)^2 = p^2 - 2pq + q^2 = p + q - 2pq$$

since for both 1 and 0, $p^2 = p$.

Many somewhat difficult determinations of logical truth values can be quickly and directly calculated using the arithmetic of one and zero.

Interpretation 1, classes, is the best interpretation for introducing Boolean algebra, and making the ideas clear. Interpretation 2, on-off circuit elements, and Interpretation 6, truth values of propositions, are the two interpretations of Boolean algebra which are the most important for applications to the design, construction, and understanding of intelligent machines. Many illustrations will appear in later chapters.

Chapter 7

INTELLIGENT MACHINES

1. Do Intelligent Machines Actually Exist?

If we agree on the meaning of intelligence, and if we accept the dictionary definition of intelligence that was quoted in the preface,

> the ability to learn from experience, to acquire and retain knowledge, and to respond quickly and appropriately to new situations, thus resulting in success in the performance of tasks,

then there is really no doubt that intelligent machines do actually exist.

The best proof however of a general statement of the type "there actually exists something or other" is to display the something or other which fulfills the requirements. Thus the best proof that the moon does sometimes eclipse the sun is to be present at an eclipse and see it happen—see a black moon entirely blot out the sun, and see darkness spread all over the land—as once I myself did have the good fortune to see, at Willimantic, Connecticut, in 1925. So, let us look at some examples of intelligent machines.

2. An Example of an Intelligent Machine

A simple example of an intelligent machine, and an example that everybody is familiar with, is a traffic light,—not the stupid kind that turns red or green according to a timetable with no regard for the traffic around it, but the kind which is able to detect the approach of traffic, and which turns red or green with some degree of appropriateness to the traffic approaching.

Two blocks from my home is a slightly intelligent traffic light, at the intersection of Otis St. and Lowell Ave. There is much more traffic on Lowell Ave. than there is on Otis St., and so the light is normally green for Lowell Ave. and normally red for Otis St. But there are two treadles on the roadway of Otis St., for cars approaching on Otis St. from either direction. If a car on Otis St. drives up to the intersection, then ordinarily, and irrespective of any traffic on Lowell Ave., the light will turn green practically at once for the Otis St. driver, and he will be able to go through the intersection with a green light and hardly any delay. But that green light for Otis St. lasts only long enough for two or three cars to drive through the intersection, and then it once more turns red. Now

if another car approaches on Otis St., that light stays red a long time, irrespective of any cars on Lowell Ave. to use it. I have timed it to last about two minutes, and then once more it will turn green for the Otis St. driver.

It is evident that this machine is not very intelligent and it is clear that it could be made much more intelligent if treadles were installed to detect traffic on Lowell Ave. But just the same, it is not nearly so annoying as a traffic light which would cycle in fixed periods, paying no attention whatever to the actual amount of traffic from moment to moment.

Now let us compare this particular traffic light with each requirement in the definition of intelligence.

Is this machine able to learn from experience and acquire and retain knowledge? Yes, in a small way, it does: it is able to take in information and hold it and distinguish between different pieces of information on the basis of time; for example, it distinguishes between the isolated car approaching on Otis St. and a second car that comes close after a first one. This distinction depends on memory.

Does the machine respond quickly and appropriately to new situations, and is it successful in its task? Yes, decidedly—even if you can deceive it some of the time by stamping your feet on a treadle. And although this particular traffic light is only moderately intelligent, we know that there are more complicated traffic lights, which employ many treadles in different places near an intersection, and which are so well engineered to the problem of regulating traffic at that intersection that they will pass more traffic through it with less delay than any policeman.

3. How Does a Machine Retain Knowledge?

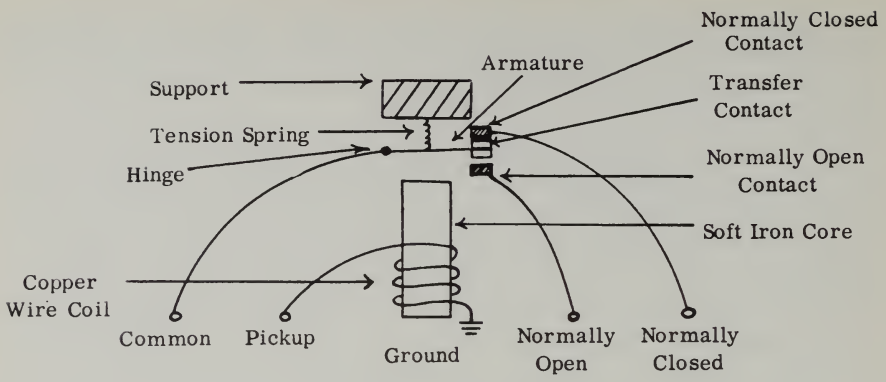
How does a traffic light actually retain knowledge? It usually retains knowledge (or remembers, or stores information) by means of electrical relays. A *relay* in general is any one of a number of devices which:

(1) holds either one of two states corresponding to "such and such event has happened" or "such and such event has not happened", and

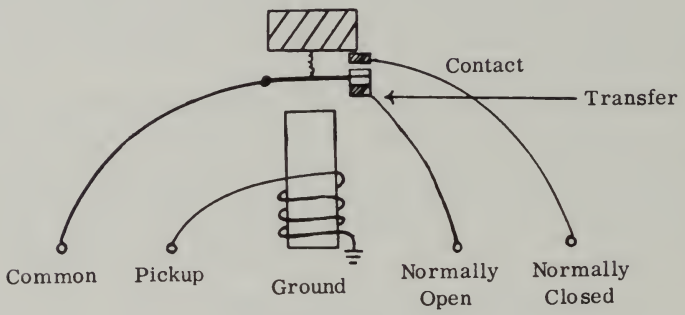
(2) is able to convey or transfer or *relay* that information about the event.

An *electrical relay* is sketched in Figure 7-1.

When no electric current flows in the coil of copper wire from Pickup to Ground around the soft iron core, the relay is not energized, and the tension spring pulls the piece of iron called the Armature up against the Normally Closed Contact, and any electric current or impulse coming from the terminal, called Common, is routed to the terminal called Nor-



NOT ENERGIZED



ENERGIZED

Figure 7-1. An Electrical Relay

mally Closed. But when electric current flows in the coil of copper wire around the soft iron core, the core becomes a magnet; it overwhelms the spring, and pulls down the armature, and presses the Transfer contact against the Normally Open contact. As a result, the transfer contact has been moved from one side, the Normally Closed side, to the other side, the Normally Open side, and any electric current or impulse from the terminal Common is routed to the terminal Normally Open.

Now it is very clear that a single relay cannot store any great quantity of knowledge or information. It cannot store a book or a paragraph or a sentence or even one decimal digit 0 to 9. But, since a relay can be energized or not energized, it can store a yes or a no, a "true" or a "false", the fact that a given event has happened or has not happened. This is precisely the basic unit of information called a *bit*, or a *binary digit*. And combinations of relays can store more information. If we use 1 for yes, or energized, and 0 for no, or not energized, two relays (read

from left to right) can store 00, 01, 10, or 11. Three relays can store any one of eight pieces of information 000, 001, 010, 011, 100, 101, 110, 111. And so on. And because n relays can store any one of 2^n pieces of information, even a small number of relays enables a great deal of information to be stored.

We have explained how an electric relay can store and transfer information. But many other kinds of physical equipment can also store and transfer information. It is not at all difficult to make physical equipment that can store the fact that a given event has happened or has not happened. But, to be able at will to transfer or relay that information easily to other physical equipment—that takes ingenuity.

A number of kinds of physical equipment—not only electrical relays—can be used for this purpose. One important kind is an electronic tube, which can be either conducting or not conducting. Another important kind is a small area on a magnetic surface (on a long plastic tape permeated with magnetic particles; or on a cylindrical drum coated with a magnetic surface; etc.); this spot can be magnetized either north-south or south-north. All these kinds of hardware are in the general sense relays: not only can they store information, but with suitable apparatus they can transfer the information.

4. Reasonable Operations on Information

But no one will maintain that the ability to store and relay information is by itself sufficient so that a device possessing this ability can properly be called intelligent. For if this were so, then a piece of paper with writing on it—something which clearly stores information and also transfers or relays it—could be called intelligent. Something more is needed: what is it?

The additional ability that must be present before a device or machine can be called intelligent—or even slightly intelligent—is the ability to do something which makes sense, to do something which is reasonable. The device needs to be able to operate automatically with information in such a way as to come out with reasonable results; it needs to be able to perform reasonable operations with information, all by itself. It must not require a human being standing at its elbow, or looking over its shoulder, providing it with all the answers.

In the case of the traffic light, at the corner of Otis St. and Lowell Ave., the apparatus inside the machine is actually able to take in the sense impressions it receives from the treadles in the pavement, and convert them by means of circuits or mechanisms into the opening and closing of the switches for the red and green lights, so that the traffic light actually functions so as to guide traffic.

Since the traffic light actually exists and functions in such a way, we have complete proof that there exists apparatus inside it that does operate reasonably with information—even if we find it at first difficult to guess how that apparatus works. We will come back at a later point to the question of the circuits by means of which the traffic light actually controls its signals. But the ability of the machine to take in information, operate upon it reasonably, and put out answers is evident and entirely undeniable; this entitles the machine to be called intelligent—at the very least, slightly intelligent.

5. Types of Intelligent Machines

The slightly intelligent traffic light is not the only machine which can take in information, store it, transfer it, operate reasonably upon it, and give out information. Other machines can perceive, accept instructions, remember, reason, calculate, decide, and give out answers.

The types of intelligent machines—machines which can operate automatically with information and perform reasonable operations—as of the present time are these:

(1) *Specific Machines (machines devoted to a specific task in a specific industry or field)*

Air traffic control equipment (including equipment on the ground), which can guide the approach of aircraft to a landing field, which can take in information about the location of aircraft in flight and give out information or control signals for the guidance of aircraft in flight.

Control systems in factories, chemical plants, oil refineries, and elsewhere, which take in indications of flow, temperature, pressure, volume, etc., in the case of liquid or powdered materials and indications of the locations and position of solid objects, the availability of machines, etc., and which give out the settings of valves, tension arms, rollers, cutting tools, etc., all according to a program of control set into the system by punched tape or other means.

Fire control equipment, which takes in indications of the presence, location, path, and speed of targets, using optical or radar perception, and which puts out times and directions for aiming and firing of guns, according to a program which calculates the motion of the target, the motion of the firing vehicle, the properties of the air, the resistance and temperature of the air, etc.

Flight simulators or training simulators, which will take in simulated conditions of flight in aircraft and the actual behavior of airplane crew members being trained, and show the necessary consequences, for the purpose of training airplane crews in "mock-ups" on the ground before they have to solve the problems in reality in the air, etc.

Game-playing machines, in which a machine will play a game with a human being, either simple games such as tit-tat-toe or nim (which have been built into some machines), or more complicated games such as checkers, billiards, or end games in chess (which have actually been programmed on the type of machine known as large-scale automatic digital computers—mentioned below).

Machine tool control equipment, which takes in a program of instructions equivalent to a blueprint, or a small size model, or the pattern of operations of an expert machinist, and puts out motions of control screws and wheels of a machine tool, so that a piece of material is shaped exactly in accordance with the instructions.

Navigating and piloting systems, which will take in star positions, time signals, radio bearings, indications of motion of the air, indications of inertia, etc., and deliver steering directions.

Network analyzers, which take in information representing measurements of resistances, inductances, and capacitances of an electrical power plant's network of electrical lines and loads, and which will enable the behavior of the system to be simulated and calculated, and thereby the system can be appropriately designed and rendered safe and economical.

Railway signaling equipment, which for example enables a large railroad terminal to schedule trains in and out every 20 seconds during rush hours with no accidents and almost no delays.

Spectroscopic analyzers, which will vaporize a small sample of material, analyze its spectrum, and report the presence and relative quantities of chemical elements contained in it.

Strategy machines, which enable military officers in training to play war games and test strategies, in which electronic devices automatically apply attrition rates and growth rates to the fighting forces and industrial resources being used in the game.

Telephone equipment, including central switching offices, automatic message accounting systems, etc., which enable one subscriber to dial another subscriber and talk to him, and which will charge the proper subscriber's account with the proper amount.

Telescope-aiming equipment, which adjusts the direction of a telescope in an astronomical observatory so that it remains pointed at the spot in the heavens which an astronomer desires to study.

Toll recording equipment, which will record, check, transmit, and summarize tolls for bridges, highways, and turnpikes.

Traffic light controllers for automobile traffic, which will detect automobile traffic approaching street intersections, and will give out stop and go signals, according to a program of generally suitable responses to the traffic conditions.

Vending machines, which will take in various coins and designations of choices, and then give out appropriate change, or merchandise, or else allow someone to play a game for a certain number of plays, etc.

(2) *General Machines (machines which may be devoted to any one of many tasks in two or more different fields)*

Accounting-bookkeeping machines, which take in numbers through a keyboard and print them in columns on a ledger sheet, but are controlled by devices expressing programs of instruction, which, according to the column in which the number is entered or other indications, cause the number to be entered positively or negatively in any one of six or more totaling counters, which can be optionally printed or cleared.

Addressing machines, which take in names and addresses on metal plates or punched cards and print the names and addresses on envelopes, wrappers, etc., according to a program of optional printing, depending on notches, punched holes, or other signals on the plates or cards.

Computers, of which there are two main kinds:

Analog computers, which take in numerical information in the form of relative measurements of physical variables, perform arithmetical and mathematical operations, are controlled by a program of instructions, and give out numerical answers.

Digital computers, which take in numerical, alphabetical or other information in the form of characters or patterns of yes-noes, etc., perform arithmetical, mathematical, and logical operations, are controlled by a program of instructions, and give out answers in numerical, alphabetical, or other form.

File-searching machines, which will store a large file of information such as photographed abstracts and references of papers together with a code for each paper, and which will very rapidly search for and find the abstracts corresponding to the searching instructions.

Inventory machines, which will store a separate total for as many as ten thousand stock items, in an equal number of registers and which will add into, subtract from, erase, and report the contents of any register requested.

Punch card machines, which will take in information expressed as a pattern of punched holes in a card of standard size and shape, and which will sort, classify, copy, list, total, print, etc., performing clerical, commercial, statistical, and many other types of office work.

Reading and recognizing machines, which will scan a printed digit or letter, observe a pattern of spots or lines, route the pattern through classifying circuits, and thereby recognize the digit or letter and activate output devices accordingly.

Robots, which will take in perceptions by means of "sensing" devices, which will put out actions by means of "acting" devices, and which will correlate sensations and actions by means of circuits or mechanisms which contain one or more programs of instructions, a "thinking" apparatus.

Test-scoring machines, which will take in a test paper completed with a pencil making electrically conductive marks and will give out the score on the test.

Typing machines, which will store on a long roll of punched paper tape, sentences, paragraphs, and other information, and which will combine these sentences and paragraphs according to instructions into correspondence, form letters, orders, etc., stopping and accepting manual "fill-ins" if so instructed.

6. The Accepted Names for "Intelligent Machines"

In spite of all these types of machines which are intelligent to a greater or less degree, people do not ordinarily refer to them as "intelligent machines". Instead, the more general names which are used are these:

(1) automatic computing machinery, or computers—where "computing" has not so much the more recent meaning of "calculating" as the root meaning "thinking closely", which is derived from the Latin *putare* "to think" and *com* meaning "intensely".

(2) data-processing machinery, or data processors—where "data" refers to information, and "processing" refers to handling.

Less general names include:

automatic controllers
simulators
analyzers
calculating machines
automatic pilots

Data processors are perhaps a broader classification than "intelligent machines". The term includes machines which do almost no reasoning, and which according to our present viewpoint are therefore not "intelligent". Such machines include: (1) the "tape-to-card converter" which will take in punched paper tape and produce punched cards, the output record containing just the same information as the input record but in a different physical medium; and (2) the facsimile copier, which with a light-sensitive electronic eye looks at successive small spots on a sheet of paper, and wherever it "sees" darkness, it stimulates a stylus to mark a spot on a second sheet of paper in the corresponding location.

Converters change from one "machine language" to another "machine language". This often requires "editing", modification of the codes or symbols according to a program of instructions. As soon as editing occurs, then again we have an "intelligent" machine.

7. The Basic Principles of Intelligent Machines

Now there is no denying that such machines as those listed above do exist and do operate: they handle information reasonably; they acquire and retain knowledge; they respond quickly and appropriately to new situations; many of them either can or may learn from experience.

But how do they do it? What are the basic principles by means of which they succeed?

The first of these basic principles is a principle of organization. Almost every instance of an intelligent machine is organized in the same way, in five parts.

(1) Each such machine has one or more inputs—ways in which it can take in information;
(2) It has one or more outputs—ways in which it can put out information;
(3) It has storage, or memory—ways in which it can store information;
(4) It has a calculating capacity (also called an "arithmetic unit")—equipment by means of which it can take in two pieces of information A and B and the specification of a reasonable operation O, such as comparison, and perform that operation on A and B, and produce as result the piece of information C; and
(5) It has a control unit, which is capable of controlling and choosing between the programs of the operation which it performs. Often the control unit too is subject to instructions calculated by the machine, so that the machine can guide itself.

The second basic principle of intelligent machines is that there exist a great many varieties of hardware, physical devices and equipment, which have powers or capacities so that information can be taken in, stored, transferred, operated upon reasonably, and put out.

Of these powers of hardware, the most difficult to imagine intuitively, and be satisfied in one's mind about, is the power to operate on information reasonably—the power to calculate.

What is an example of this? An ordinary adding machine, with its paper tape to show the numbers entered, and a key for causing the machine to print a total and then clear, is undeniable evidence of at least some power to calculate. The clue to the operation of the adding machine is a device called a *counter wheel*, which is divided into ten equal

segments around the rim corresponding to the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. The counter wheel has a little extra tooth located between 9 and 0, so that when the counter wheel turns through that point, the extra tooth catches in the next counter wheel above it and causes that one to turn from any digit it stands at to the next digit. This clever little tooth, invented by the French mathematician Pascal in the 1600's, expresses in hardware the principles of arithmetical carry.

Is it really true that every kind of logical and reasoning operation, every kind of arithmetical and mathematical operation can be carried out by machine? As far as we can now tell, yes. Mathematicians and symbolic logicians have shown that vast numbers of complicated operations can be defined in terms of simpler ones, and the simpler ones can be defined in terms of still simpler ones, until finally only about half a dozen simplest operations remain. All others can be arrived at as combinations of these simplest ones. These simplest operations, and many more complicated ones, can be directly expressed in hardware as the on-ness and off-ness of signals and circuits, patterns of interaction of yeses and noes.

Chapter 8

SMALL MACHINES THAT REASON: SOME SIMPLE PROBLEMS

1. The Behavior of Machines

A machine that is to behave in a certain desired way can often be imagined as a "black box", as some undefined thing filled with hardware in some indefinite way that nevertheless has the very definite property that certain inputs produce certain outputs. For example, most people have only a very vague idea of how a telephone works, but they all know that if you do certain things to a telephone, you can get certain things out of it. The same is true of a motor car, for another example: we simplify it in our thoughts into a black box, a something, which has certain inputs and certain outputs.

What are the main inputs to a motor car instant by instant? The steering wheel: how much you turn it. The gas pedal: how far you push it down. The brake pedal: how far you push that down.

What are the main outputs of a motor car instant by instant: Speed: how fast it goes. Direction: which way it goes.

If we imagine abstractly this behavior of a motor car, we can think of it as like the "black box" drawn in Figure 8-1:

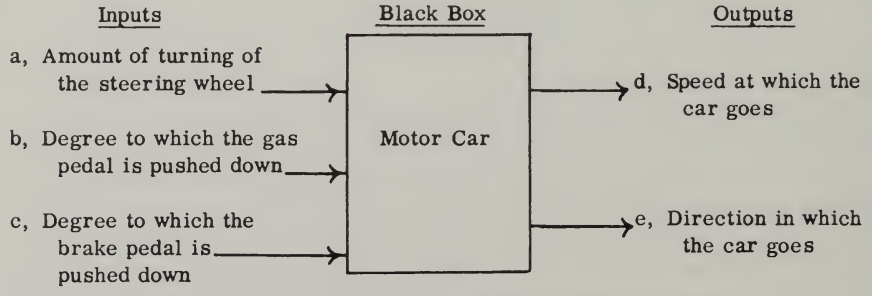


Figure 8-1. Schematic of the Behavior of a Motor Car

The drawing shows physical variables *a*, *b*, *c*, *d*, *e*. These physical variables may be reported in numbers: *a*, in degrees plus or minus measured away from straight forward; *b* and *c*, distances in inches or fractions

of an inch; *d*, in miles per hour; and *e*, in degrees plus or minus measured away from straight forward; *d* and *e* are mathematical functions of *a*, *b*, and *c*. For example, I have been told that *e* is usually one quarter of *a*, except that in a Volkswagen car, *e* equals *a*. But the important thing is that our use of a motor car depends on our knowledge of how *d* and *e* are determined, once we establish *a*, *b*, and *c*.

This same general pattern, a "black box" with three inputs and two outputs, fits another machine; a lighting circuit that has the following properties:

Inputs: Hall switch, that can be up or down
Upstairs switch, that can be up or down
Attic switch, that can be up or down
Outputs: Porch light, that may be on or off
Garage light, that may be on or off

We can think of the lighting circuit schematically as shown in Figure 8-2:

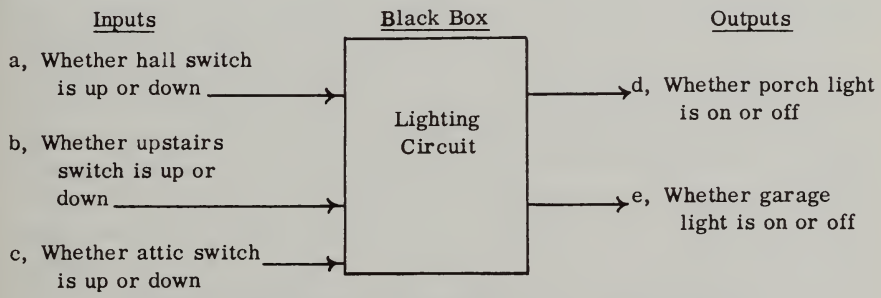


Figure 8-2. Schematic of the Behavior of a Lighting Circuit

But there is an important difference between these two machines: for the motor car, the variables *a* to *e* are measured on a scale and reported in ordinary numbers; for the lighting circuit, however, the variables *a* to *e* have only two values corresponding to the two states up or down, on or off. For reporting them, it is convenient to make use of the Boolean algebra of 1 and 0, the algebra of truth values (on-off elements).

We find on-off elements or yes-no elements in countless places in machines (and also elsewhere in the world). A switch may be open or closed. A button may be pressed or not pressed. A key of a machine may be down or up. An indicator of floors in an elevator may show a flag or not show a flag. A wire may be carrying current or not carrying current. An electrical relay may be energized (current flowing through the pick-up coil) or not energized (current not flowing through the coil). An

electronic tube may be conducting current (a flow of electrons) or not conducting. It is useful and convenient to call all these types of elements *on-off elements*, and those that occur in circuits *on-off circuit elements*.

Any on-off element E in a mechanism or a circuit corresponds to a statement P "The element E is on". The statement has a truth value $T(P)$ equal to p. When the statement is true, it has the truth value p equal to 1; and we associate 1 with the element. When the statement is false, it has the truth value 0; and we associate 0 with the element. Since most of the time we want to cover both cases, both the case "on" and the case "off", we associate the variable p with the element E. E is a particular piece of hardware; P is a statement mentioning it and saying that it is "on"; and p is a variable which may have either 1 or 0 as its values. We can call p a *binary variable*. This is a general approach, and is therefore powerful.

It is often convenient to think of p as the state of the circuit element, or as the information or signal contained in the circuit element. In the same way, for an element that could be at any position n along a scale, we would say that n is the setting or numerical value of that element.

For example, if the element E is a switch, the statement P may be "the switch is closed", and p equals 1 if it is closed, and 0 if it is not closed. If the element is a relay, the statement is "The relay is energized". If the element is a wire, the statement may be "The wire is carrying current". If the element is the plate of an electronic tube, the statement may be "The voltage of the plate is high", where we agree that we are interested in just two voltages "high" and "low". And so on, and so forth, for an unlimited number of different kinds of on-off circuit elements.

Now, the behavior of a machine is specified when we describe all the possible values of the input variables, and all the possible values of the output variables, and describe how they are related. Understanding the behavior of a machine means understanding this relation.

2. Switches

The on-off circuit element which will be the most interesting to us is the electrical switch. It has many forms. It is much more widely applicable than any mechanical on-off element, because a switch works remotely at the ends of wires or other electrical connectors. Therefore, the spatial restrictions of three dimensions which are severe upon mechanical elements, and which require clever inventors for their solutions, do not affect switches.

A switch was originally a device for switching a train from one track to another. Now in addition it is a device for turning an electrical current from one path to another.

In Figure 8-3, the parts of an ordinary switch are shown: the *transfer* or *common wire* and terminal; the *hinge* by means of which the *leaf* of the switch may turn; and the *normally closed* terminal and wire and the *normally open* terminal and wire, to either of which the leaf of the switch may be connected. In Figures 8-4 and 8-5 are shown more schematic, less detailed drawings of a switch.

A switch may have three or four or more *positions*, and in such a case the positions may have any convenient names (Figures 8-6, 8-7, 8-8 and 8-9).

A single switch may be constructed so that it has two or three or more electrically nonconnecting sections (often called *decks* or *poles*) so that as the switch is turned, it simultaneously switches two or more electrically independent paths. In circuit diagrams, this property of a switch may be shown by using a name for the switch and numbers 1, 2, 3, etc. for the decks. For example, a switch (named Z) with three positions (A, B, and C) and four decks (named 1, 2, 3, and 4) is diagrammed in Figures 8-9 and 8-10.

Many other on-off circuit elements are equivalent in one way or another to a switch. For example, an electrical relay is an automatically operated switch, where an electromagnet is used to transfer the leaf from one contact to another, instead of human fingers. Its fastest switching speed is about one thousandth of a second. Two electronic tubes arranged so that one conducts electricity when the other does not, and vice versa, is the equivalent of a switch and the usual switching speed of this tube pair is on the order of a few millionths of a second.

3. Designing Small Machines that Reason

Now let us narrow the discussion of all machines and their behavior to the subject of small machines that reason; and let us include the subject of how Boolean algebra is used to design them. For it is an interesting and important fact that the set of statements that describe the behavior of a small machine that reasons, when translated into Boolean algebra, becomes the key to the pattern or design or arrangement of on-off circuit elements that will make the machine operate. We shall illustrate this in a moment.

What do we mean by *small machines that reason*? These are machines whose inputs and outputs consist of binary variables, and the pattern of

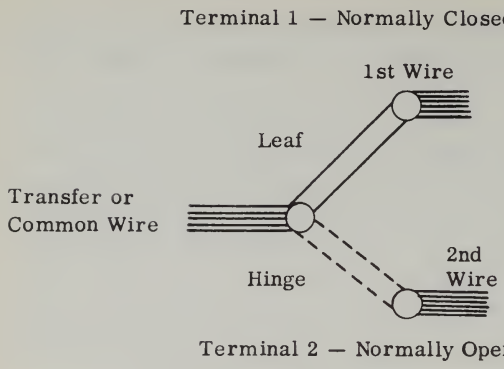


Figure 8-3. Switch

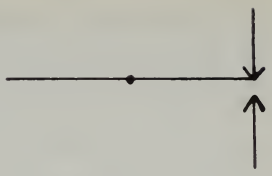


Figure 8-4. Switch Schematically Drawn

Figure 8-5. Switch Schematically Drawn

Figure 8-6. Three-Position Switch

Figure 8-7. Four-Position Switch

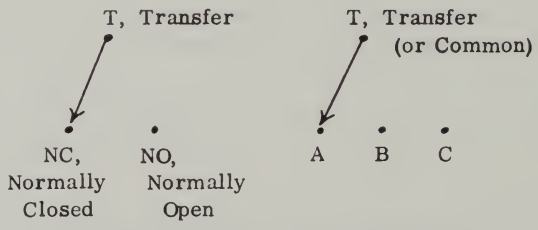


Figure 8-8. Names and Designations of Contacts

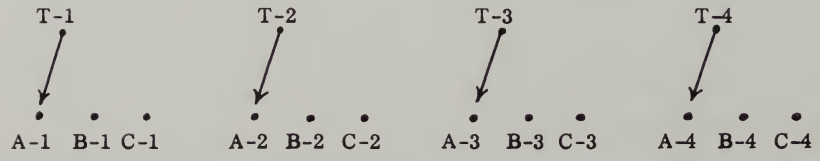


Figure 8-9. Four-Deck Three-Position Switch

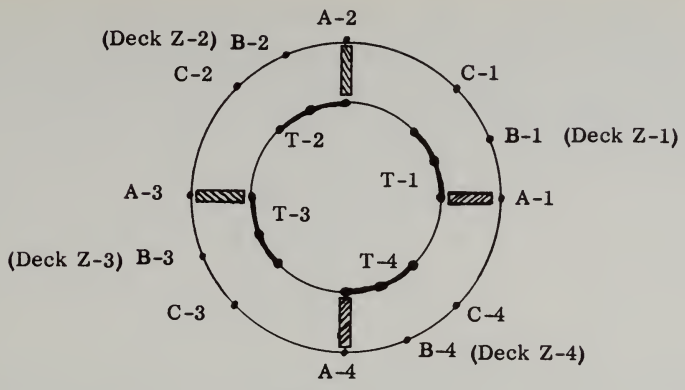


Figure 8-10. Four-Deck Three-Position Switch Z

1's and 0's in the outputs is a mathematical function of the 1's and 0's in the inputs.

Let us take a simple example and examine it.

4. The Problem of the Hall Light

Problem: An electrical circuit is to be installed in a house. There will be two switches, a Downstairs Switch and an Upstairs Switch. There will be one light, the Hall Light. When either one of the two switches is turned, the light is to go on if it was off, and to go off if it was on.

Design the circuit.

Solution: Let u be the state of the upstairs switch U, 1 if that switch is turned up, and 0 if that switch is turned down.

Let d be the state of the downstairs switch, 1 if it is turned up and 0 if it is turned down.

Let h be the state of the hall light H, 1 if it is on, and 0 if it is off.

Now the two switches can be set in four possible combinations, giving rise to cases 1, 2, 3, and 4, as shown:

Case No.	u	d
1	0	0
2	0	1
3	1	0
4	1	1

Now, how shall we meet the requirement of the problem?

Well, suppose that in Case 1 the hall light is off, h is 0. Then if either switch is turned by itself, the light should go on. Therefore, h for cases 2 and 3 should be 1. Finally, if the other switch is turned, resulting in case 4, the light should go off again. So we can complete the table as follows:

Case No.	u	d	h
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Looking at the table we can sum up what we see there by saying: h is 1 if and only if u is 1 and d is not 1, or u is not 1 and d is 1. In symbols: $h = ud' \vee u'd$

So we can make a circuit which looks like this:

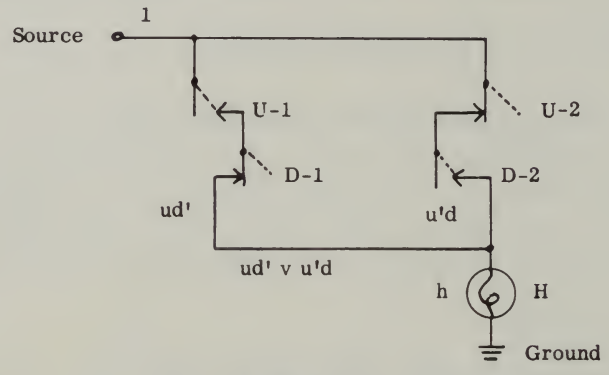


Figure 8-11

In this circuit, we use two decks of Switch U and two decks of Switch D. The only signal which can pass through the left-hand path is ud', because the U transfer contact has to move over, and the D transfer contact has to stay where it is drawn. The only signal that can pass through the right-hand path is similarly $u'd$. So the only signal that can reach the light H is $ud' \vee u'd$, and hence the circuit is correct.

But have we made the best use of the hardware? We are not using the normally closed terminal for deck U-1, and we are not using the normally open terminal for deck U-2.

If we experiment for a short while with switches and their Boolean equations, we can find a better circuit; it has the same equation but uses less hardware; it is shown in Figure 8-12. Now we can get along with less expensive switches, one deck each instead of two decks each.

This kind of simplification is often possible, within limits. The expressions of Boolean algebra give one solution, and then the particular properties of the switches (manual, electromechanical as in a relay, or electronic as with a pair of tubes) enable the solution to be improved from the point of view of producing the same result but using less equipment.

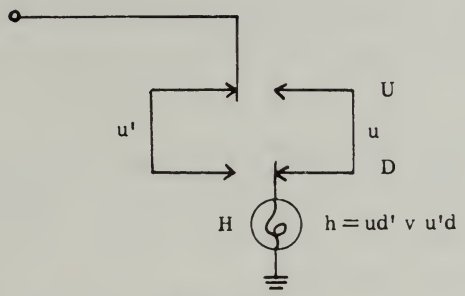


Figure 8-12

5. The Problem of Bruce Campbell's Will

Now let us take a slightly more complicated problem about a small machine that reasons, and solve it.

Problem: Here is Bruce Campbell's will:

If at my death my son, Bruce Campbell II, is not living, and if no son of his and grandson of mine bearing the name Bruce Campbell III, is then living, then 40% of my estate will be paid to the heirs of my son. If my son is living at my death, and is not a graduate of Edinburgh University and is not married, and has no son living named Bruce Campbell III, then 40% of my estate will be paid to my son. If my son is living, but is a graduate of Edinburgh University or is married, but has no son living named Bruce Campbell III, then 60% of my estate will be paid to my son. If my son is living and has a son living named Bruce Campbell III, then my son will get all of my estate if he is a graduate of Edinburgh University but only 80% of my estate if he is not a graduate. If my son is not living but if he had a son named Bruce Campbell III who is living at my death, then 80% of my estate will be paid to Bruce Campbell III or his legal guardian. Any balance of my estate will be paid to the Gaelic Home for the Aged and Indigent.

How much of Bruce Campbell's estate is paid to his son or his son's heirs?

How design a machine which will indicate at once what is paid, for any combination of conditions?

Solution: In this problem there are four conditions: We can conceive of the class of all possible happenings which would be classified by these four conditions, so that we can use the Boolean algebra of classes.

son is Living (L) or not Living (L') (1)

son is Married (M) or not Married (M') (2)

son is a Graduate of Edinburgh University (G) or not a Graduate (G') (3)

a specially Named grandson, Bruce Campbell III, is living (N) or not living (N') (4)

The first sentence in the problem is:

"If at my death my son, Bruce Campbell II, is not living, and if no son of his and grandson of mine bearing the name Bruce Campbell III is then living, then 40% of my estate will be paid to the heirs of my son". We can translate this sentence as:

IF (Son not living) AND (Named grandson not living), THEN 40%.

In symbols:

$$L' \cdot N' \longrightarrow 40\% \tag{5}$$

The second sentence, similarly translated, becomes:

IF (Son living) AND (Son not a graduate) AND (Son not married) AND (Named grandson not living), THEN 40%.

In symbols, this becomes:

$$L \cdot G' \cdot M' \cdot N' \longrightarrow 40\% \tag{6}$$

The third sentence translated becomes:

IF (Son living) AND [(Son a graduate) OR (Son married)] AND (Named grandson not living), THEN 60%.

In symbols, this becomes:

$$L\ (G \vee M)\ N' \longrightarrow 60\% \tag{7}$$

The fourth sentence translated becomes:

IF (Son living) AND (Named grandson living) AND (Son a graduate), THEN 100%;

IF (Son living) AND (Named grandson living) AND (Son not a graduate), THEN 80%.

In symbols this becomes:

$$L \cdot N \cdot G \longrightarrow 100\% \tag{8}$$

$$L \cdot N \cdot G' \longrightarrow 80\% \tag{9}$$

The fifth sentence translated becomes:

IF (Son not living) AND (Named grandson living), THEN 80%.

In symbols this becomes:

$$L' \cdot N \longrightarrow 80\% \tag{10}$$

Next we check to see if the problem has been properly stated. Is the set of conditions mutually exhaustive and exclusive? Conditions (6) and (7) have no cases in common since $(G'M') \cdot (G \vee M) = 0$; and together they come to:

$$LG'M'N' \vee L(G \vee M)\, N' = LN' \tag{11}$$

Conditions (8) and (9) have no cases in common since $G \cdot G' = 0$; and together they come to:

$$L \cdot N \tag{12}$$

Conditions (5), (10), (11) and (12) have no cases in common, and together they add to the universe class:

$$L'N' \vee L'N \vee LN' \vee LN = 1$$

Hence the problem has been properly stated, and the set of conditions is mutually exhaustive and exclusive; and therefore it is possible to design a consistent machine which will display the results.

Now, how do we design the machine?

The machine will have four switches. One switch will be labeled "Grandson Named Bruce Campbell III Living", and it will have two positions labeled "Yes" and "No". The second switch will be labeled "Son Living" and it will have two positions labeled "Yes" and "No". The third switch will be labeled "Son a Graduate" and it will have two positions labeled "Yes" and "No". The fourth switch will be labeled "Son Married"; it will have two positions labeled "Yes" and "No". There will be four output lights, and they will be labeled 40%, 60%, 80%, and 100%.

How do we know that the machine will have this structure? Well, first, the machine has to be able to take in the report "yes" or "no" about each one of these four conditions. A convenient way for a small machine to take in information "yes" or "no" is to set the position of switches. Secondly, the machine has to be able to put out any one of four possible answers. A convenient way to signal different answers that a machine produces is to signal with lights.

What about the structure of the machine between the input and the output? Of course there are more than dozens of ways for designing this

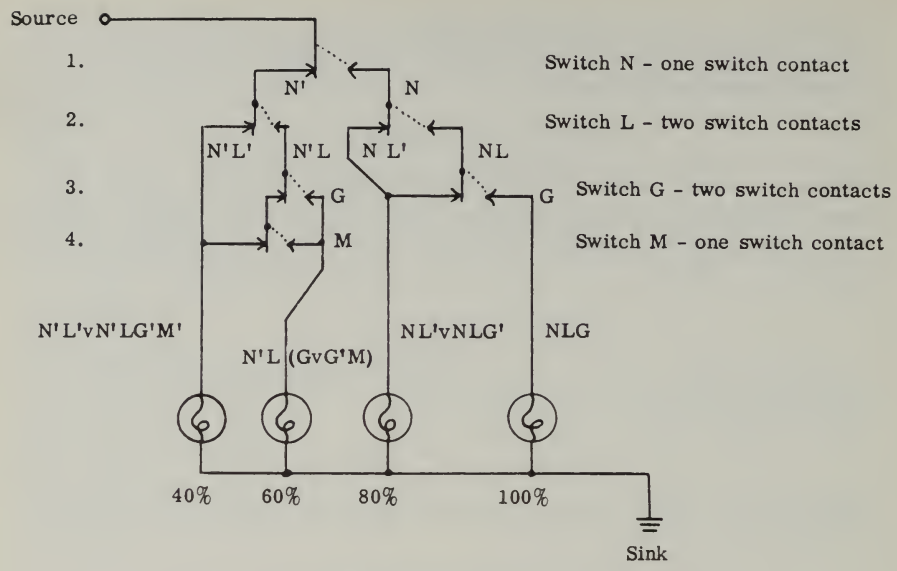


Figure 8-13. Machine for Bruce Campbell's Will

structure; in Figure 8-13 is one of them, an actual design for the machine, a circuit diagram using electrical hardware. Let us trace out the circuit.

The small circle o at the top of the diagram designates a source of current; the symbol at the bottom of the diagram designates a sink of current, or ground. If you wish, you may think of these as one side and the other side of an electric battery. Now let us think of electric current as a flow of small marbles of electricity (electrons) along hollow tubes drawn as lines (actually wires) shown in the figure.

The electrons start at the source; then they come to a fork in the path, at Level 1 in the diagram, the level marked "Switch *N*—one contact". The switch is drawn in the "No" position. Suppose this switch is in the "No" position; then the small marbles of electricity will run down the left hand side of the leaf of the switch, through the contact point marked with an arrow head, and down into the adjacent left-hand wire. This wire is marked with the information it contains, *N'*, "Named grandson not living". Contrariwise, now suppose the switch is moved and placed in the "Yes" position instead. Then the flow of electrons will not go straight down, but will flow diagonally along the dotted path of the leaf (the transfer contact), and then flow through the contact point marked with an arrow head down through the adjacent right-hand wire. This wire is marked with the information it contains, *N*, "Named grandson living".

We now come to Level 2 in the diagram. Both of the switch contacts in this row are switch contacts on the same switch *L*. They are electrically isolated; but they move in unison, so that if either contact moves, the other moves also. As a result the four lines that issue from the contacts of Switch *L*, carry four different items of information, respectively, $N'L'$, $N'L$, NL', and NL, from left to right.

Now when we compare these pieces of information with the output of the problem, we see that $L'N'$ leads to or implies 40%, and that $L'N$ implies 80%. So we can connect these two lines without further fussing to two of the output lights. But we must now further classify information contained in the other two lines, by making further use of switch contacts of the third switch *G* and the fourth switch *M*. One contact of switch *G* is sufficient to select the 100% cases; the line *N* AND *L* AND *G* (GNL) implies 100%. LNG' implies 80%, and can be connected to the 80% light.

Two contacts, one of Switch *G* at Level 3 and one of Switch *M* at Level 4 are used to produce $G'M'$ along one line and $G \vee G'M$ (which is the same as $G \vee M$) along the other line, so that finally we have $N'L' \vee N'LG'M'$ running to the 40% light and $N'L(G \vee G'M)$ running to the 60% light.

Chapter 9

SMALL MACHINES THAT REASON: A SYLLOGISM MACHINE, AND OTHER PROBLEMS

1. The Problem of the Syllogism Machine

Now let us take a problem which comes directly out of ordinary logical reasoning, the problem of making a machine which will reason correctly about syllogisms. What is a syllogism?

A *syllogism,* according to the dictionary, is a logical scheme for analyzing a formal argument, which has these properties: (1) it consists of two statements called *premises,* and a third statement called a *conclusion;* and (2) the conclusion necessarily follows from the premises—in other words, if the two premises are true, the conclusion must be true.

One of the most famous examples of a syllogism is: "All men are mortal. Socrates is a man. Therefore Socrates is mortal." The same form of argument appears in: "All birds have feathers. That hawk up in the sky is a bird. Therefore that hawk has feathers." The structure of the reasoning appears in the words and framework: "All .... are ----. That :::: is a .... . Therefore, that :::: is a ----". The structure also appears in: "All *B*'s are *C*'s. That *A* is a *B*. Therefore that *A* is a *C*." The *A*'s, *B*'s, and *C*'s are called *terms.*

In this kind of reasoning, for our purposes, we can disregard the difference between singular and plural. A class that contains a single element and a class that contains more than one element do not need to be distinguished. For example, money can be thought of in the plural as a collection of pieces of money, or it may be thought of in the singular as a collective entity like water, or gravel, or people. So we can also say that the same structure of argument appears in: "All *B* is *C*. All *A* is *B*. Therefore all *A* is *C*."

The word syllogism also has a second meaning. It also denotes a pattern of reasoning that looks like or seems to be a syllogism, but is not correct logically. The reason is that people have to have a name for a pattern of reasoning that looks like a syllogism, and it may take some calculating to find out if it is actually true or not true.

The two premises and the conclusion of a syllogism often have one of the following forms:

First Premise	Second Premise	Conclusion
1. All *A* is *B*	5. All *B* is *C*	9. All *A* is *C*
2. No *A* is *B*	6. No *B* is *C*	10. No *A* is *C*
3. Some *A* is *B*	7. Some *B* is *C*	11. Some *A* is *C*
4. Some *A* is not *B*	8. Some *B* is not *C*	12. Some *A* is not *C*

These are not all the forms of premises and conclusions in a syllogism, but they cover a large part of reasoning using syllogisms.

Now it is of course false that a combination at random of premises and conclusion is likely to be correct reasoning. For example, if we choose statements 2 and 7, we cannot deduce statement 9. In other words from "No *A* is *B*. Some *B* is *C*" we cannot deduce that "All *A* is *C*". It is easy to eliminate some of the fallacies. But other fallacies are often maintained vigorously by people who should know a great deal better. For example, the former Senator from Wisconsin, Joseph McCarthy, maintained "All Communists attack me. So-and-so attacks me. Therefore so-and-so is a Communist". The fact that this claimed "syllogism" is false can be seen by comparing it with "All caterpillars eat lettuce. I eat lettuce. Therefore I am a caterpillar." But the actual logical proof that such a syllogism is false is not regularly taught to a student in American schools until he finally takes a college course in logic or philosophy.

Now let us examine a problem.

Problem: Design a machine which will take in any one of Statements 1 to 4 as one input, and any one of Statements 5 to 8 as a second input, and which for output will indicate any one of statements 9 to 12 (or more statements if needed) if such statement is a *correct* conclusion.

If we can solve this problem, we shall have a machine which will automatically analyze 16 syllogisms—showing what conclusion can be drawn, if any can be drawn, and showing also if true, "No conclusion about *A* and *C* is possible".

We can solve this problem, as will be shown in the solution displayed below. The resulting "syllogism machine" is a distinctly powerful example of a small machine that logically reasons. In fact, the machine is three times an example: (1) it is an example of how logical reasoning can be performed by a machine; (2) the machine is an example of how a reasoning circuit can be designed by using Boolean algebra; and (3) the problem that it expresses is an example of Boolean algebra calculation.

Solution: The machine consists of two input switches, each with four positions, and an output consisting of four or more lights which may be on or off. One switch is labeled "First Premise"; it can be set at any one of the four positions:

(1) All *A* is *B*
(2) No *A* is *B*
(3) Some *A* is *B*
(4) Some *A* is not *B*

The second switch is labeled "Second Premise"; it can be set at any one of the four positions:

(5) All *B* is *C*
(6) No *B* is *C*
(7) Some *B* is *C*
(8) Some *B* is not *C*

The first four lights are labeled:

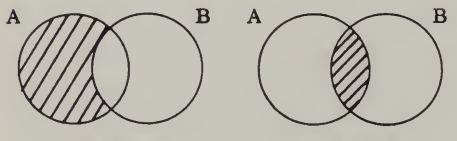
(9) All *A* is *C*
(10) No *A* is *C*
(11) Some *A* is *C*
(12) Some *A* is not *C*

It turns out that three more lights are needed. They are labeled.

(13) Some *C* is not *A*
(14) Some not-*A* is not-*C*
(15) No conclusion about *A* and *C* is possible

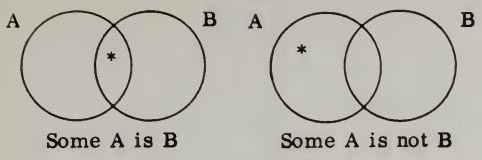
Making this machine involves two problems. The first problem is to use Boolean Algebra to draw all the conclusions that we can from the 16 possible combinations of the first premise and the second premise. The second problem is to use Boolean algebra to express all the 16 answers in the form of a circuit for the machine.

The easiest way to display Boolean algebra in working out the conclusions from each of the 16 possible cases is to make use of Venn diagrams. Venn diagrams express some of the relations of two classes as follows:



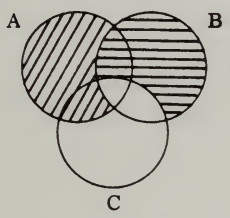
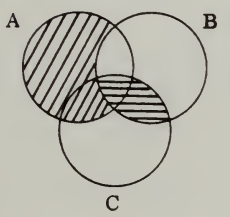
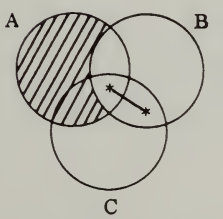
All A is B No A is B

Crosshatching an area shows that it is empty.

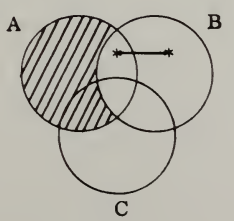


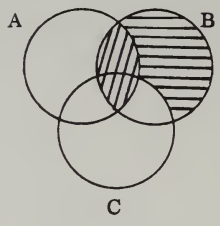
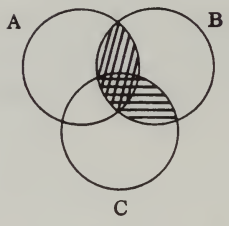
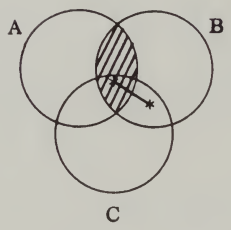
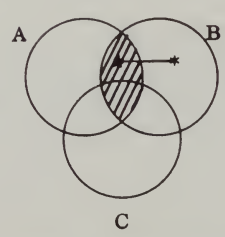
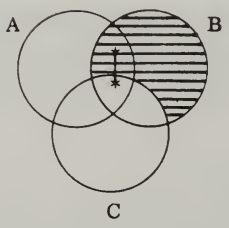
Putting an asterisk in an area shows that it is not empty.

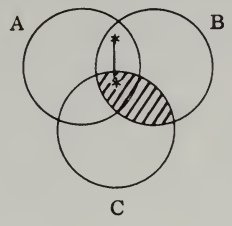
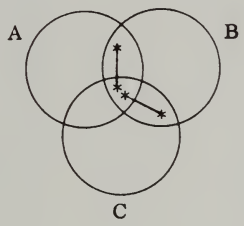
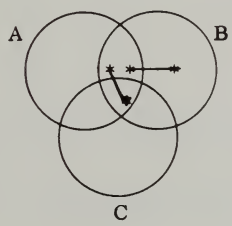
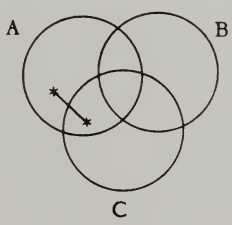
Here is the calculation of the 16 cases:

Case	Venn Diagram	In Words
1,5		All *A* is *B* All *B* is *C* Therefore, all *A* is *C* (Light 9)
1,6		All *A* is *B* No *B* is *C* Therefore, no A is *C* (Light 10)
1,7		All *A* is *B* Some *B* is *C* Therefore, no conclusion about *A* and *C* is possible (Light 15)

Note that we do not know the location of the asterisk on one side or the other of the A-Class boundary line. Consequently, we draw two asterisks connected by a line crossing the boundary, to represent the lack of knowledge.

1,8		All *A* is *B* Some *B* is not *C* Therefore, no conclusion about *A* and *C* possible (Light 15)

Case	Venn Diagram	In Words
2,5		No *A* is *B* All *B* is *C* Therefore, no conclusion about *A* and *C* is possible (Light 15)
2,6		No *A* is *B* No *B* is *C* Therefore, no conclusion about *A* and *C* is possible (Light 15)
2,7		No *A* is *B* Some *B* is *C* Therefore, some *C* is not *A* (Light 13)
2,8		No *A* is *B* Some *B* is not *C* Therefore, some not-*A* is not-*C* (Light 14)
3,5		Some *A* is *B* All *B* is *C* Therefore, some *A* is *C* (Light 11)

Case	Venn Diagram	In Words
3,6		Some *A* is *B* No *B* is *C* Therefore, some *A* is not *C* (Light 12)
3,7		Some *A* is *B* Some *B* is *C* Therefore, no conclusion about *A* and *C* is possible (Light 15)
3,8		Some *A* is *B* Some *B* is not *C* No conclusion about *A* and *C* is possible (Light 15)
4, and any of 5,6,7,8		Some *A* is not *B* (Any) Therefore, no conclusion about *A* and *C* is possible (Light 15)

We now come to the problem of designing the circuit. We make up a table of the cases. A table of cases, by the way is one of the commonest and easiest methods of carrying out Boolean algebra and other logical reasoning.

Here is the table of cases:

First Premise	Second Premise	Output
1	5	9
1	6	10
1	7	15
1	8	15
2	5	15
2	6	15
2	7	13
2	8	14
3	5	11
3	6	12
3	7	15
3	8	15
4	any	15

This table of cases may be converted into an expression of Boolean algebra as follows. Note that the numbers are not used in any arithmetical sense at all, but only as names of statements or names of circuit elements, in other words, as elements of Boolean algebra.

$9 = 1(5)$ $\quad$ $13 = 2(7)$
$10 = 1(6)$ $\quad$ $14 = 2(8)$
$11 = 3(5)$ $\quad$ $15 = 1(7 \vee 8) \vee 2(5 \vee 6) \vee 3(7 \vee 8) \vee 4$
$12 = 3(6)$

The scheme of the circuit therefore is as shown in Figure 9-1. Electrically, in order to obtain the four decks of the switch for the Second Statement, we use different sectors of one and the same multiple switch.

Another Boolean algebra expression for this table of cases is:

$$9 = 5(1)$$
$$10 = 6(1)$$
$$11 = 5(3)$$
$$12 = 6(3)$$
$$13 = 7(2)$$
$$14 = 8(2)$$
$$15 = 5(2 \vee 4) \vee 6(2 \vee 4) \vee 7(1 \vee 3 \vee 4) \vee 8(1 \vee 3 \vee 4)$$
$$= (5 \vee 6)(2 \vee 4) \vee (7 \vee 8)(3 \vee 4)$$

This expression would result in a different circuit that would accomplish the same result.

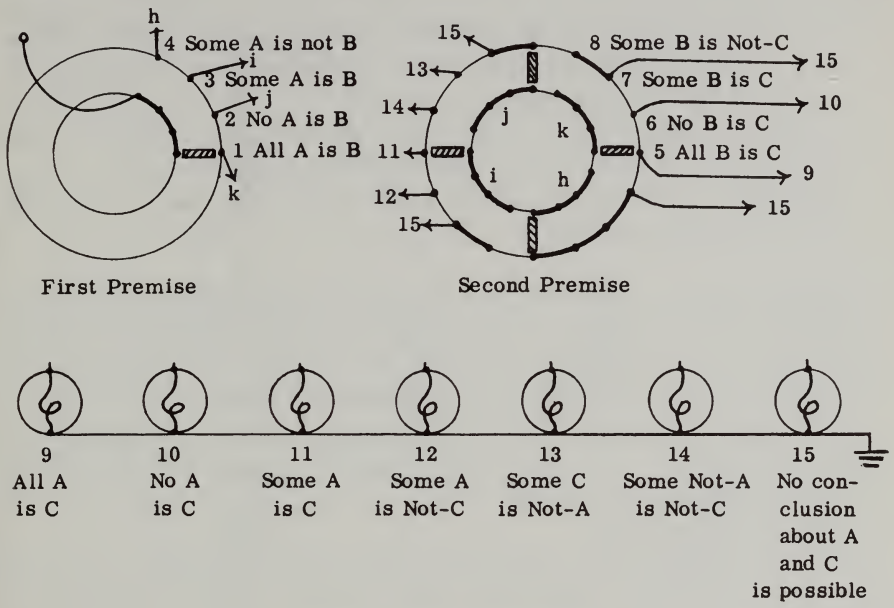


Figure 9-1. Syllogism Machine.

2. The Problem of the Logical Truth Calculator

The preceding section has shown us a machine which, given two premises, will determine the conclusion. Can we carry this kind of machine for logical reasoning further? Can we construct a machine which will calculate the logical truth or falsity of combinations of statements?

The answer is yes. To show that this is true, let us consider two problems, taken from the collection called Brainiacs (see page 193).

First Problem: Sandy Campbell, an electrical engineer, has begun to take a course in the algebra of logic at Edinburgh University with Professor Higgins. One day Professor Higgins says to his students that there are seven important logical operators and connectives for statements, and that they can be expressed using very common and familiar English words. The Professor explains, "If we let *P* stand for a statement, and if we let *Q* stand for a statement, then there are at least the following seven important combinations:

(a) *P* AND *Q* (in the sense *P* and *Q* are both asserted to be true)
(b) *P* OR *Q* (in the sense that *P* or *Q* or both are asserted)

(c) NOT-P (in the sense that the statement P is denied)

(d) P OR ELSE Q (in the sense that either P or Q is true but not both of them)

(e) IF P, THEN Q (in the sense of asserting that Q is true or P is false, or both; or, equivalently, in the sense of denying that P is true and Q is false)

(f) P IF AND ONLY IF Q (in the sense of asserting that P is true and Q is true or else P is false and Q is false)

(g) NEITHER P NOR Q (in the sense of asserting that P is not true and Q is not true)

Professor Higgins says "For each of these combinations, if you know whether P is true or false, and if you know whether Q is true or false, then you can decide absolutely whether the combination is true or false, because of your knowledge of what the connective means."

Sandy gets to thinking about this after he gets home, and a few days later he brings into class seven small machines which demonstrate just what the professor was saying.

Design these machines.

Solution: For each of these machines except NOT-P (c) there will be two switches; one for the statement P and the other for the statement Q. Each switch will have two positions, TRUE and FALSE. There will be two lights: RESULT TRUE, RESULT FALSE.

The information which the circuits must express is shown in the following table of truth-values:

P	Q	(a) P AND Q	(b) P OR Q	(d) P OR ELSE Q	(e) IF P, THEN Q	(f) P IF AND ONLY IF Q	(g) NEITHER P NOR Q
0	0	0	0	0	1	1	1
0	1	0	1	1	1	0	0
1	0	0	1	1	0	0	0
1	1	1	1	0	1	1	0
		$p \cdot q$	$p \vee q$	$pq' \vee p'q$	$q \vee p'$	$pq \vee p'q'$	$p'q'$

P	NOT-P
0	1
1	0
p	p'

The Boolean equation which each of these tables of truth values expresses is shown at the bottom of the table. These are the equations by means of which we can solve the problem and design circuits for the machines. These circuits are shown in Figure 9-2.

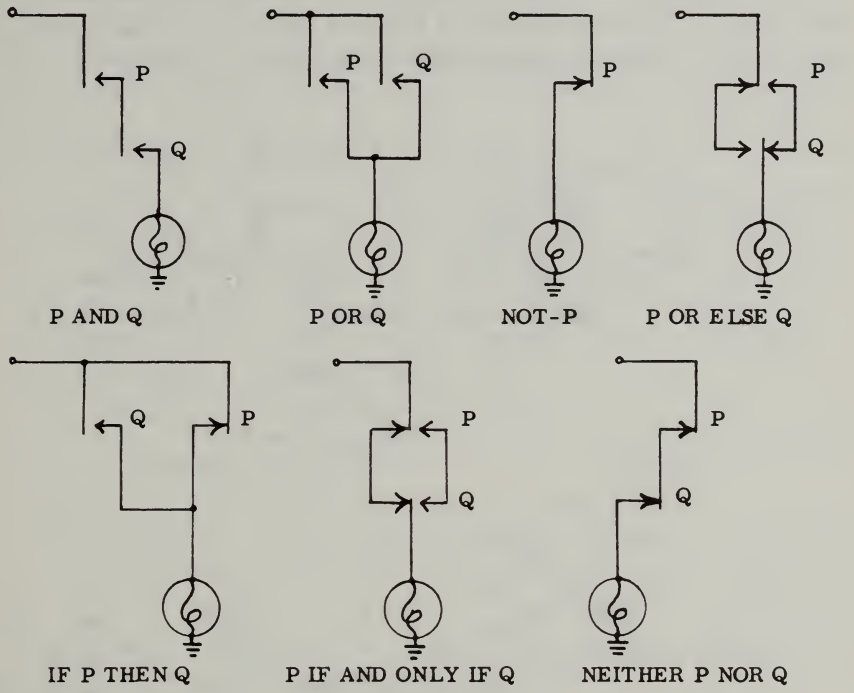


Figure 9-2

Second Problem: Another student in Professor Higgins' class, after listening to the presentation by Sandy Campbell the following morning, and Professor Higgins' comments, raises his hand and says: "It seems to me that all these machines except the machine for NOT-*P* could be combined into one machine, with a switch for choosing the connective or operator". Professor Higgins replies "You are right. Two young men named T. Kalin and W. Burkhart did exactly this some years ago. They made a machine with room for putting in any one of ten statements, and for selecting any one of four connectives between pairs of statements or combinations of pairs and then exploring the possible combinations of true and false. For next time, since you suggested the idea, why don't you bring to class a small machine which will demonstrate selecting the connective as well as the truth value of *P* and *Q*?"

Design a machine which will meet this assignment.

Solution: There will be three switches, CONNECTIVE, 1ST STATEMENT *P*, and 2ND STATEMENT *Q*. The Connective Switch will have the positions AND, OR, OR ELSE, IF ...THEN, IF AND ONLY IF, and NEITHER ...NOR. The other two switches will have the positions TRUE and FALSE. There will be two RESULT lights, one marked TRUE, the other FALSE. The circuit is shown in Figure 9-3.

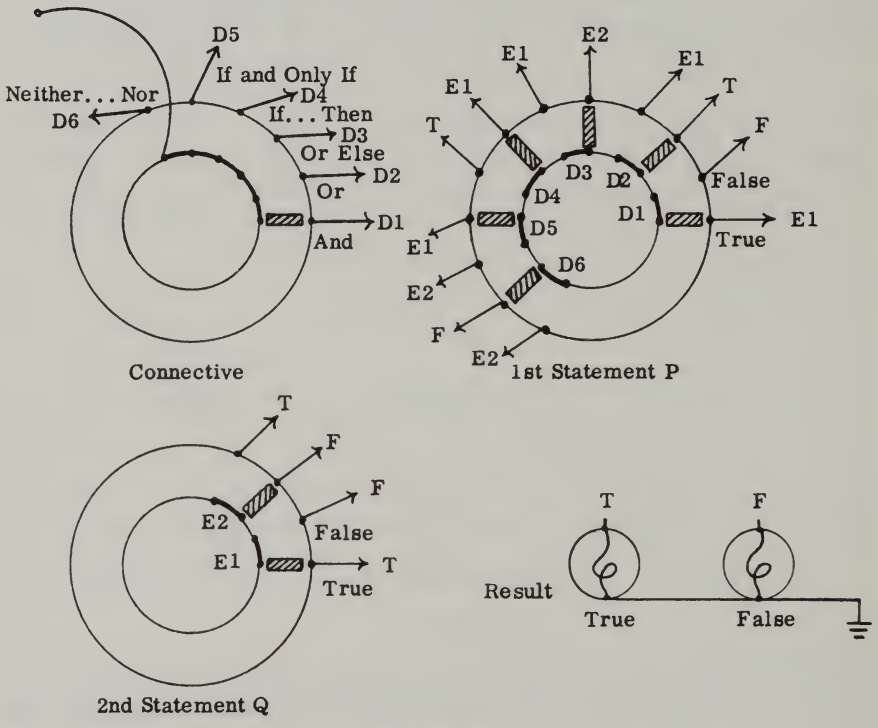


Figure 9-3

3. The Problem of the Magazine Editor's Argument

The last two problems have been a little abstract; let's consider a problem in logical reasoning which is more concrete, and which though modified is based on a situation which once actually occurred in the author's business.

Problem: The managing editor, Mr. Robinson, said to his assistant, Mr. Jones: "The story on Dutch Harbor is set in 1/2 page columns, and if we put its continuation next to that 2/3 page ad, it will have to be reset by the printer. If the printer resets it in 1/3 page columns, he will make

mistakes. If he makes mistakes, we shall have no chance to catch them because the magazine deadline is very close. Then there will be mistakes in the final issue and we shall look foolish. We do not want to look foolish, and so we should put the continuation of the story about Dutch Harbor somewhere else where resetting in 1/3 page columns is not necessary."

Mr. Jones distrusts a good part of this argument. He wants a machine which will show under what conditions the editorial staff will look foolish, or not.

The statements that appear in this argument are:

(1) The story on Dutch Harbor is set in 1/2 page columns.
(2) If its continuation is put next to a 2/3 page ad, then the continuation will have to be reset by the printer in 1/3 page columns.
(3) The printer will make mistakes in typesetting the continuation.
(4) The editorial staff will have no chance to catch the mistakes.
(5) The magazine deadline is very close.
(6) There will be mistakes in the final issue.
(7) If there are mistakes in the final issue, the editorial staff will look foolish.
(8) The editorial staff does not want to look foolish.
(9) The editorial staff puts the continuation of the story somewhere else where resetting in 1/3 page columns is not necessary.

Mr. Jones is entirely willing to accept statements (1), (2), (7), and (8), but he would like to analyze the possible logical alternatives among the remaining statements with a logical machine.

Design the machine.

Solution: The relation among the remaining statements (3), (4), (5), (9) as premises and (6) as conclusion are:

If not-9, and if 3, and if 5, and if 4, then 6.
If 9, or if not-9 and if not-3, or if not-9 and if 3 and if not-5, or if not-9 and if 3 and if 5 and if not-4, then not-6.

This relation can be expressed in a circuit. There are four switches: PRINTER MAKES MISTAKES (3); EDITORIAL STAFF WILL HAVE NO CHANCE TO CATCH MISTAKES (4); MAGAZINE DEADLINE IS VERY CLOSE (5); and EDITORIAL STAFF PUTS CONTINUATION ELSEWHERE (9). Each switch has two positions YES and NO. There will be two lights: YES and NO, reporting on the statement MISTAKES IN FINAL ISSUE AND EDITORIAL STAFF LOOKS FOOLISH (6 and 7). (See Figure 9-4.)

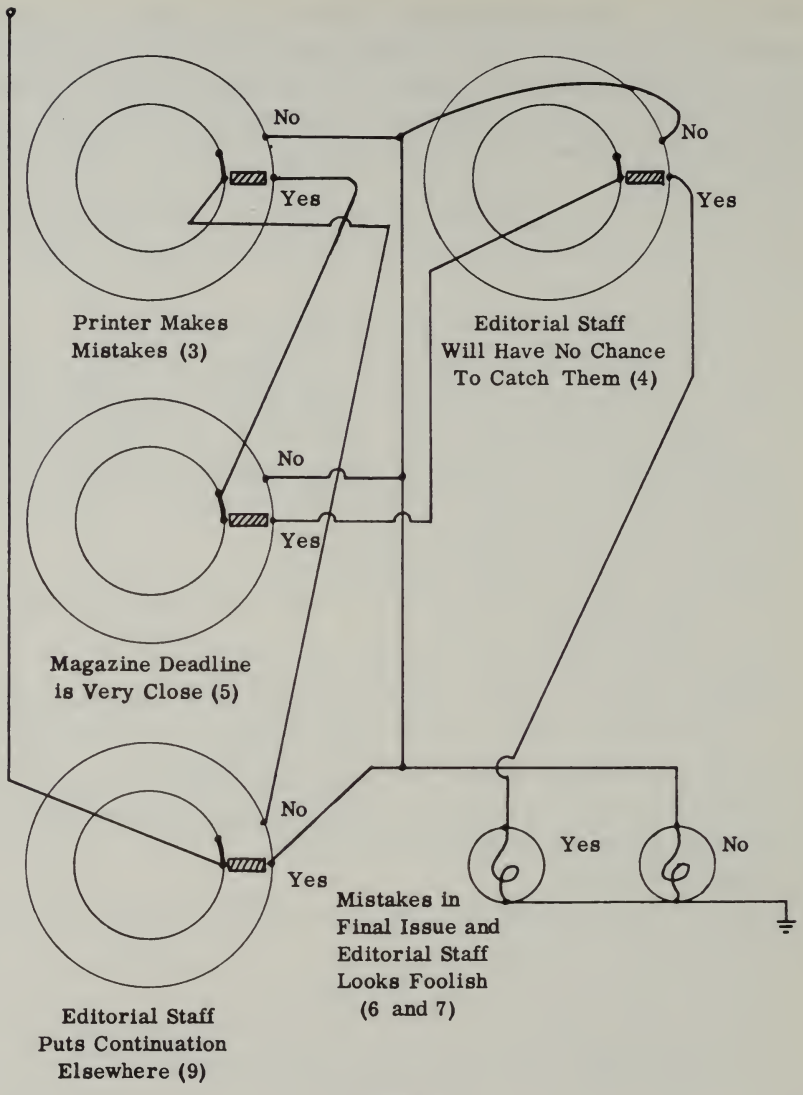


Figure 9-4

Note: In only one of sixteen possible settings of the four switches is there a YES light (Mistakes in final issue, and editorial staff looks foolish). In the other fifteen possible cases there is only a NO light. Hence there is some ground for Mr. Jones' suspicion of the argument, since it relies on the truth of four out of four statements for it actually to apply.

4. Do These Machines Really Reason?

After an engineer has designed one of these small machines that operate reasonably with information, and after a technician has put it together so that it works, does such a machine really reason?

It cannot be successfully denied that the machine really reasons, and reasons more correctly and faster than a human being.

But, you may say, the small machine does not understand the reasoning process, for no knowledge of the reasoning process has been built into its circuits.

True. And a similar statement is true for a human being. The human being does not know how the reasoning process goes on in his own brain. A human brain is more inaccessible to investigation than the top of Mount Everest. The part of the human brain that carries out reasoning is embedded somewhere in a mass of nerves and tissues and blood vessels in complete darkness somewhere inside the skull; it at once ceases to operate if anybody tries to look at it, take it apart, test it, observe it physically—and no human being can put it back together again. In fact, we are more likely to find out how reasoning goes on in our own brains from construction of models of reasoning machines made out of hardware, than we are by direct investigation of small parts of the brain.

You may say that the small machine cannot invent a reasoning process or any part of it.

True. But a machine large enough and powerful enough should theoretically be able to invent a reasoning process.

And again, a similar statement is true about a human being: no single human being has invented the reasoning process that human beings use. The process is the outgrowth of six thousand years of language and culture and civilization, six thousand years of thought and calculation and small inventions of reasoning steps and small inventions of devices (like writing and the abacus) to aid the flighty, inadequate memory of the primate, shown clearly in a great many human beings, chimpanzees, and other apes and monkeys.

In fact it should not be hard to prove that no one human being could have invented the reasoning process, because the amount of abstraction and the amount of mental work required to do so is definitely far beyond the powers of a single human isolated from a favorable culture.

And for the future, we may confidently expect that small machines that reason will be here, there, and everywhere, in schools and colleges, homes and businesses, so that human beings will be helped both in learning the reasoning process and in gaining the results from accurate reasoning by machines.

Chapter 10

LARGE MACHINES THAT COMPUTE: THE PROBLEM OF ADDING DECIMAL DIGITS

The principles described and illustrated in the last chapter can also be applied to machines that *compute*, that perform arithmetical calculations as well as logical calculations. One reason that this is true is that arithmetical calculations can be analyzed rather easily into logical calculations, making particular use of the *binary number system* or binary system of notation for numbers. The binary system has as digits only 0 and 1, and the places or columns of a number denote successive powers of two. For example, the number 1101 in binary notation stands for one eight (which is two cubed) plus one four (which is two squared) plus no two (which is two to the first power) plus one unit (which is two to the power zero), or in total thirteen. By contrast, in the decimal notation, the number 1101 stands for one thousand (which is ten cubed) plus one hundred (which is ten squared) plus zero tens (which is ten to the first power) plus one unit (which is ten to the zero power).

A machine which is to do useful computing, needs to be considerably larger than a machine which only makes logical calculations. The reason for this is that the amount of information which needs to be handled, the number of ones and zeros representing the arithmetical information, is considerably larger in arithmetical calculation than in logical calculation. For, most of the time, the numbers we wish to calculate with are numbers from about three up to about eight decimal digits; and even six decimal digits may require 19 binary digits to express them. This requires additional equipment or additional time or both.

1. The Decimal Addition Table in Hardware

To show the reasons for some of these statements, let us choose for an example the basic problem of expressing in hardware the addition table in the decimal system. A machine, if it is to add in the decimal system, must either have this table built into its hardware or have some equivalent of this table built into its hardware.

The collection of facts which we need to express in the hardware of a machine is shown in Table 10-1:

TABLE 10-1. ADDITION TABLE IN THE DECIMAL SYSTEM

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

As we observe the smooth and even regularity of the numbers in this table, it is not surprising to realize that this collection of 100 addition facts was first expressed in hardware by small revolving gears or dials, with ten equally spaced teeth around their circumference. If such a dial stood at 4, and if the stroke of a latch or ratchet pushed it through the space of 3 gear teeth, then the dial would stand at 7, and addition in one column would have been accomplished. If the next stroke of the latch or ratchet pushed the dial through the space of 5 gear teeth, then the dial would stand at 2; and while the dial passed through the position from 9 to 0, a little extra side tooth on the dial would nudge the gear next to it, making it advance one tooth, thus "carrying a 1" into the dial recording the position of the adjacent gear. This system is three hundred years old, and is the foundation of almost all mechanical desk adding and calculating machines.

2. The Decimal Addition Table Expressed in Ten-Position Switches

We can try to express this kind of hardware element with 10 positions in electrical switches, in a circuit which is a close relative to some of the circuits in the preceding chapter.

Problem: Design a machine which will take in any digit *A* from 0 to 9 on a ten-position switch, Switch A, and which will take in any digit *B* from 0 to 9 on a second ten-position switch, Switch B, and which will give out the sums, which will, of course, range from 0 to 18. Such a machine will express the addition table in the decimal system.

Solution: Switch A will be a ten-position switch with one deck, and Switch B will be a ten-position switch with 10 decks. There will be 19 Sum lights S0 to S18. The circuit is shown in Figure 10-1.

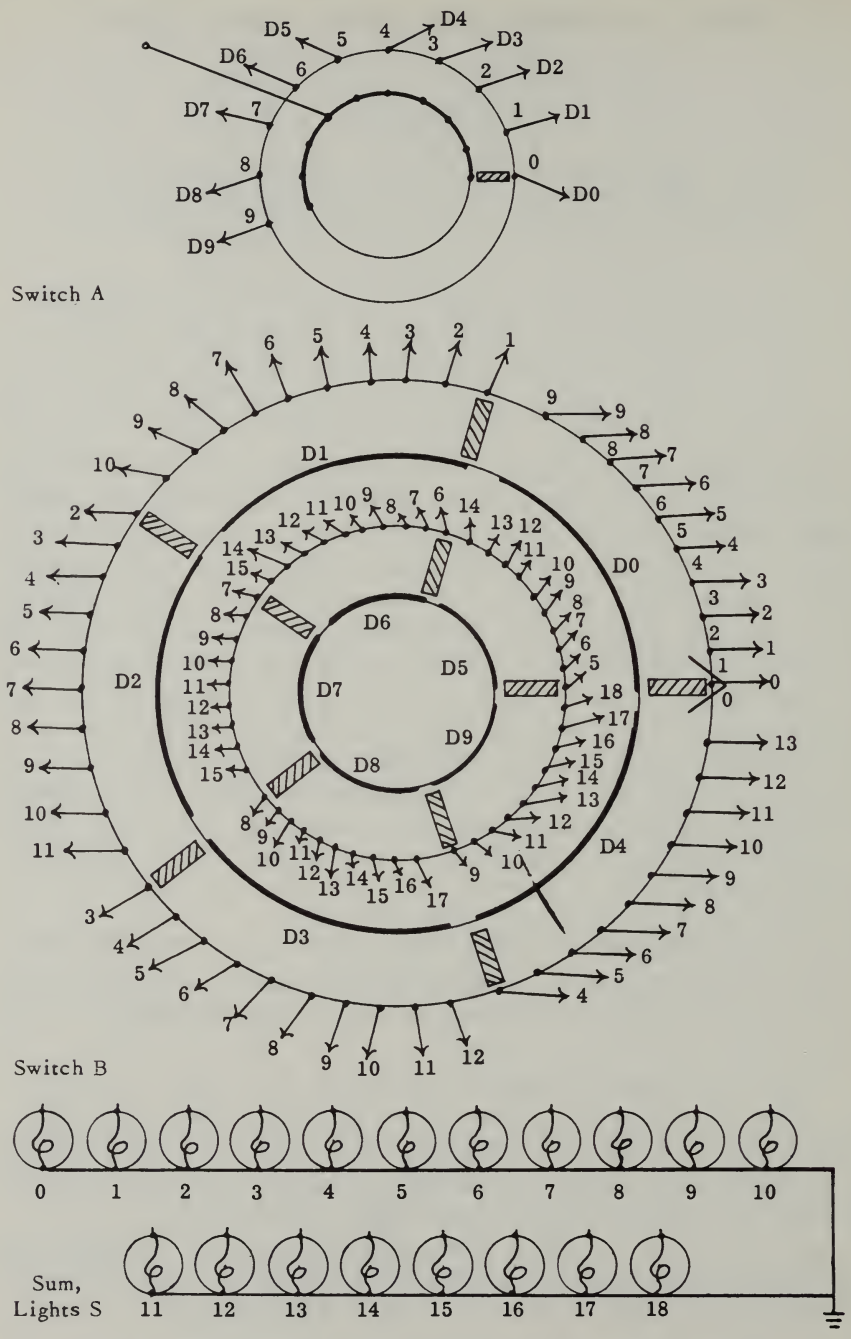


Figure 10-1. Decimal Addition Table—A Preliminary Circuit

It expresses simply and straightforwardly the following series of Boolean equations:

$$
\begin{aligned}
S0 &= A0 \cdot B0 \\
S1 &= A0 \cdot B1 \vee A1 \cdot B0 \\
S2 &= A0 \cdot B2 \vee A1 \cdot B1 \vee A2 \cdot B0 \\
S3 &= A0 \cdot B3 \vee A1 \cdot B2 \vee A2 \cdot B1 \vee A3 \cdot B0 \\
&\ldots\ldots \\
S9 &= A0 \cdot B9 \vee A1 \cdot B8 \vee A2 \cdot B7 \vee \ldots\ldots \vee A9 \cdot B0 \\
&\ldots\ldots \\
S18 &= A9 \cdot B9
\end{aligned}
$$

3. Criticisms

One glance at this circuit, however, shows that, even though this small machine is a solution to the problem, it is not a good solution. In the first place, this machine is not economical in hardware; surely we don't need 121 terminals and 121 wires for the solution. Secondly, we know we shall have to deal with many more digits than a single one, just as soon as we pass from the case of numbers of one digit to the case of numbers of 8, 10, or more digits. Thirdly, we must be able to take the result of one addition, given in the output of the machine, and transfer it back into an input of the machine, so that the same machine can be used over and over again, in fact for an indefinitely large number of additions.

According to the experience which has been gained in the technique of automatic computers, the thing to do is abandon the use of mechancial or electrical elements which have 10 different and distinct positions, such as the dials or switches that we have been talking about. They display marked disadvantages when we try to greatly increase machine speed. They are slow. The fastest ten-position mechanical elements can add two numbers in about a fifth of a second, when one number is already in the accumulating dial register of the machine, and the other number has already been entered in the keyboard register, waiting for the addition button to be pressed. This is of course no disadvantage when a human being is operating the machine, because the human being takes very much longer than a fifth of a second for entering a number into the machine or copying a number out of the machine. But as soon as we seek to arrange automatic control over a long sequence of calculating operations of the machine—and this is the heart of the great development of automatic data processing since 1944—then we discover that mechanical or electrical elements with 10 different positions are slow and inconvenient. It turns out that we need to change over to combinations of mechanical or electrical elements each of which has only two positions. We arrive at the binary system and its close relatives.

4. The Decimal Addition Table Expressed in Eight Two-Position Switches

Let us try to take some of these conditions into account, and make some changes accordingly in our problem of constructing a machine that will express the addition table.

First let us express each of the two decimal digits A and B which we are adding not as a single switch with ten positions but as a set of four switches each with two positions. Then we shall have each decimal digit defined as a set of four binary digits; and one pattern for this definition is shown in Table 10-2. In addition let us require that the machine give two outputs, one the left-hand decimal digit of the sum (0 or 1), and the other the right-hand decimal digit of the sum (0 to 9), this decimal digit being expressed as four binary digits also according to Table 10-2. What will the circuits be then?

TABLE 10-2

Decimal Digit	Four Binary Digits
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Problem 2. Design a machine which will take in a decimal digit A on four switches A8, A4, A2, A1, and take in a decimal digit B on four switches B8, B4, B2, B1, and which will shine a light C1 if the left-hand digit of the sum is 1, and which will shine in lights S8, S4, S2, S1 the value of the right-hand decimal digit of the sum.

Solution: The four A switches will each have two positions and the four B switches will each have two positions. The five output lights will correspond to C1, S8, S4, S2, S1.

5. The First Output S1

Let us consider the equations in Boolean algebra which will determine C1, S8, S4, S2, and S1, Let us begin with the simplest, the equation for S1.

S1 will be 1 if and only if A is even and B is odd, or B is even and A is odd. In other words,

$$S1 = A1 \cdot B1' \vee A1' \cdot B1$$

This relation is expressed in Table 10-3 and Figure 10-2.

TABLE 10-3

A	B
0,2,4,6,8	1,3,5,7,9
1,3,5,7,9	0,2,4,6,8

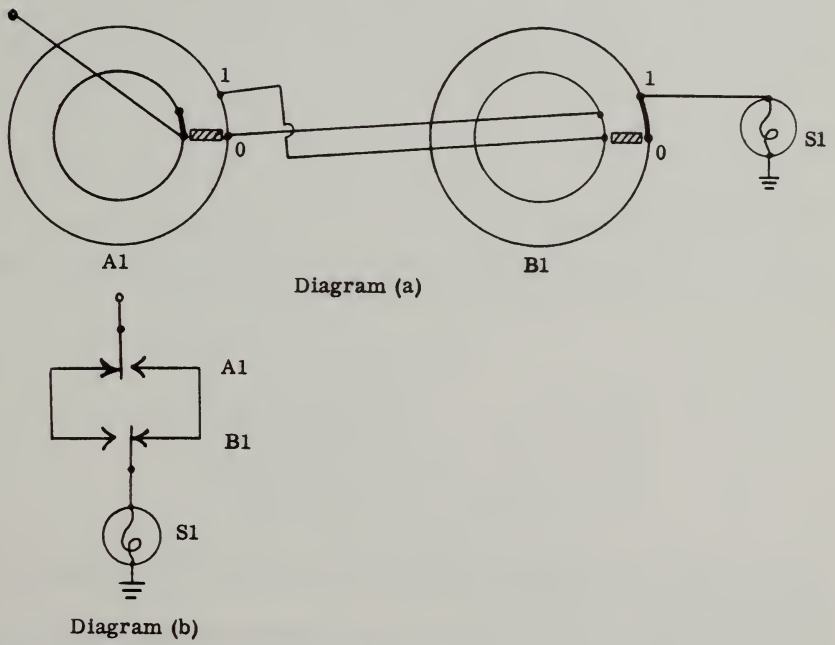


Figure 10-2

In Figure 10-2 Diagram (a) the circuit is drawn almost pictorially, assuming we have rotating two-position switches; the 0 (OFF) and 1 (ON) positions are represented by dots and names, and the transfer contact is represented by a heavy line connecting two adjacent terminals on the switch and by the jumper. In Diagram (b) the same switch is represented schematically and much more simply; the 0 contact is represented by an arrow touching (drawn in the normally CLOSED position); the 1 contact is represented by an arrow not touching (drawn in the normally OPEN po-

sition); and the transfer contact is represented by a simple line touching the arrow point for OFF. We shall use the style shown by Diagram (b) for many future circuit drawings.

6. The Second Output, C1

The condition that C1 = 1, that is, the left hand digit of the sum is one, is that *A* plus *B* is greater than or equal to ten. This will be true in the forty-five cases shown in Table 10-4.

TABLE 10-4

A	*B*	*A*	*B*
1	9	6	4 to 9
2	8,9	7	3 to 9
3	7,8,9	8	2 to 9
4	6 to 9	9	1 to 9
5	5 to 9		

All these cases and no others must be included in the circuit.

We need to express these 45 cases as a Boolean function of A8, A4, A2, A1, B8, B4, B2, B1. Let us begin by considering the cases which single Boolean conditions include. These are shown in Table 10-5.

TABLE 10-5

A8 = 8,9	A8′ = 0,1,2,3,4,5,6,7
A4 = 4,5,6,7	A4′ = 0,1,2,3,8,9
A2 = 2,3,6,7	A2′ = 0,1,4,5,8,9
A1 = 1,3,5,7,9	A1′ = 0,2,4,6,8

We can see that A8 implies A4′ and A2′, and of course therefore A4 implies A8′ and A2 implies A8′.

Grouping digits as suggested by the information in Table 10-5, we can group the forty-five cases into Boolean conditions as shown in Table 10-6.

TABLE 10-6

Group	Condition	Cases	No. of Cases
1	A8 · B8	A8,9;B8,9	4
2	A4 · B8 v B4 · A8	A4 to 7,B8,9 and B4 to 7, A8,9	16
3	A4′ · A2 · B8 v B4′ · B2 · A8	A2,3,B8,9 and B2,3, A8,9	8

TABLE 10-6 (*Continued*)

Group	Condition	Cases	No. of Cases
4	$A4 \cdot B4 \cdot B2 \vee B4 \cdot A4 \cdot A2$	A4 to 7, B6,7 and B4 to 7, A6,7	12
5	$A1 \cdot B1 (A8' \cdot A4' \cdot A2' \cdot B8 \vee B8' \cdot B4' \cdot B2' \cdot A8 \vee A4' \cdot A2 \cdot B4 \cdot B2 \vee B4' \cdot B2 \cdot A4 \cdot A2 \vee A4 \cdot A2' \cdot B4 \cdot B2')$	A1 · B9, A9 · B1, A3 · B7, A7 · B3, and A5 · B5	5

The logical sum of these conditions—the result of associating all these conditions with the Boolean operator "v"—is the total condition that the circuit for C1 should express. Using this total condition, we can (1) write down the circuit without too much trouble, and (2) prove that the circuit is correct.

Remembering that A8 implies A4′ and A2′ and that either A4 or A2 implies A8 we can write the total condition as:

$$
\begin{aligned}
C1 = {} & A8 \cdot B8' \\
& \vee A8 \cdot B8' \cdot B4' \\
& \vee A8' \cdot B8 \cdot A4' \\
& \vee A8 \cdot B8' \cdot B4' \cdot B2' \\
& \vee A8' \cdot B8 \cdot A4' \cdot A2' \\
& \vee A8' \cdot B8' \cdot A4 \cdot B4 \cdot B2' \\
& \vee A8' \cdot B8' \cdot A4 \cdot B4 \cdot A2' \\
& \vee A8' \cdot B8 \cdot A4' \cdot A2' \cdot A1 \cdot B1' \\
& \vee A8 \cdot B8' \cdot B4' \cdot B2' \cdot A1 \cdot B1' \\
& \vee A8' \cdot B8' \cdot A4' \cdot B4 \cdot B2 \cdot A2 \cdot A1 \cdot B1' \\
& \vee A8' \cdot B8' \cdot A4 \cdot B4' \cdot B2 \cdot A2 \cdot A1 \cdot B1' \\
& \vee A8' \cdot B8' \cdot A4 \cdot B4 \cdot A2' \cdot B2' \cdot A1 \cdot B1'
\end{aligned}
$$

This equation reduces to:

$$
\begin{aligned}
C1 = {} & A8 \cdot B8 \\
& \vee A8 \cdot B8' (B4 \vee B2 \vee A1 \cdot B1) \\
& \vee A8' \cdot B8 (A4 \vee A2 \vee A1 \cdot B1) \\
& \vee A8' \cdot B8' [A4 \cdot B4 (A2 \vee B2 \vee A1 \cdot B1) \\
& \qquad \vee (A4' \cdot B4 \vee A4 \cdot B4') (A2 \cdot B2 \cdot A1 \cdot B1)]
\end{aligned}
$$

This circuit can be expressed in the circuit diagram of Figure 10-3, containing 22 switch points. By considering the information in each of the lines, it can be further simplified into the circuit diagram of Figure 10-4 containing 14 switch points.

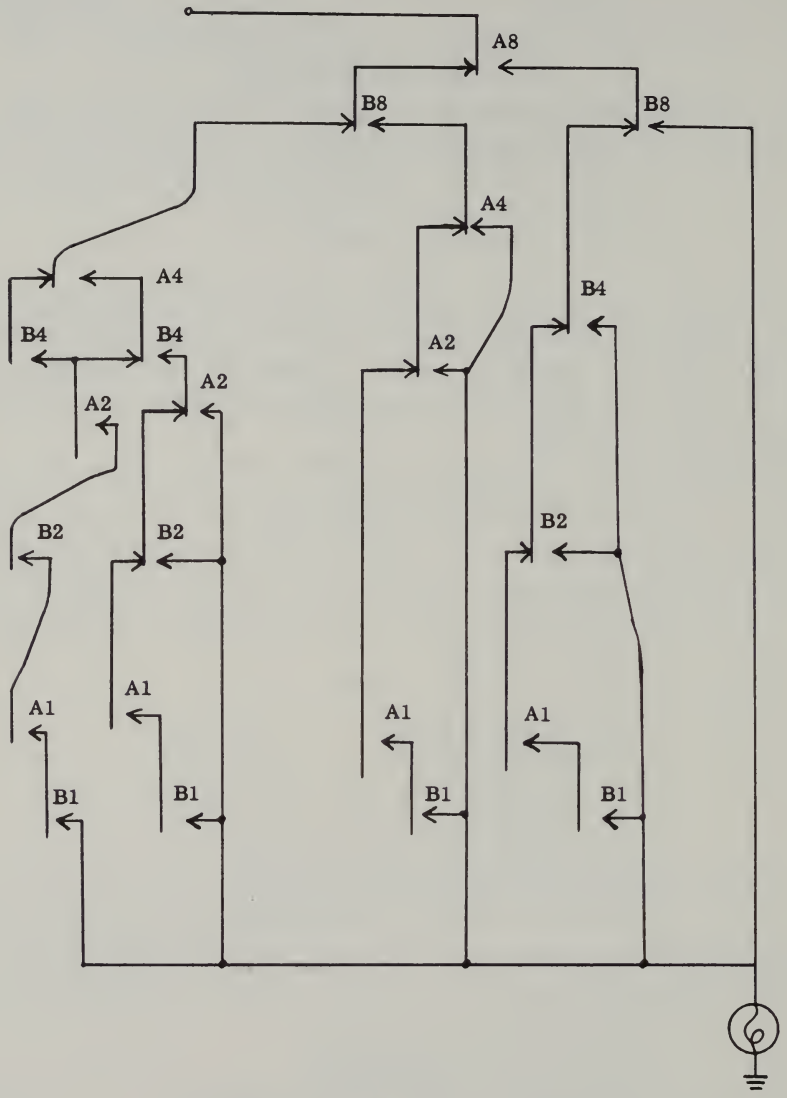


Figure 10-3

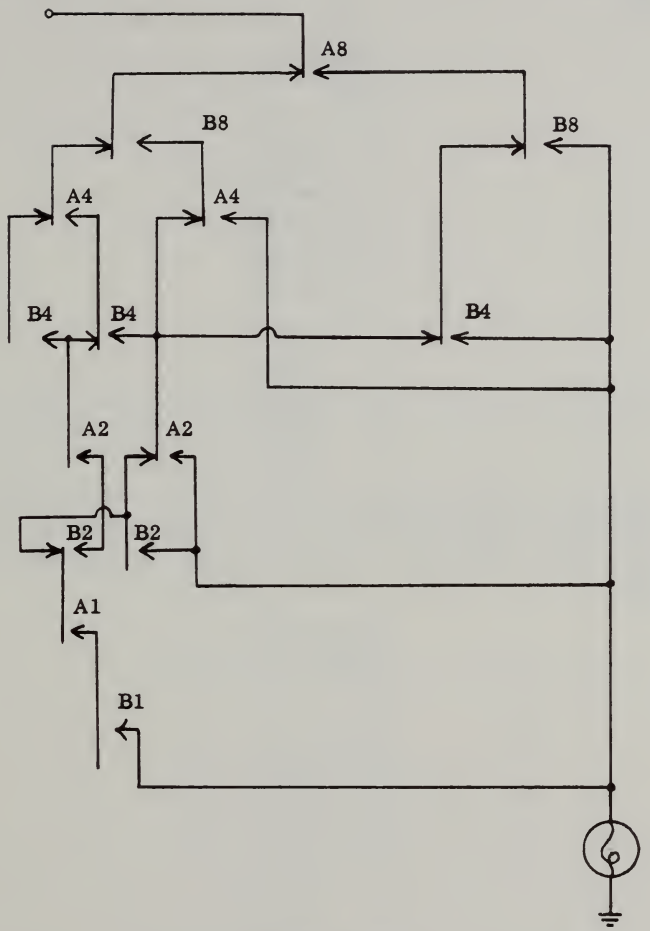


Figure 10-4

7. The Next Three Outputs S2, S4, S8

The condition that S2 equals 1 is the condition that the right hand digit of the sum is 2, 3, 6, or 7. This will be true in the cases shown in Table 10-7.

TABLE 10-7

A	*B*
0	2,3,6,7
1	1,2,5,6
2	0,1,4,5
3	0,3,4,9
4	2,3,8,9
5	1,2,7,8
6	0,1,6,7
7	0,5,6,9
8	4,5,8,9
9	3,4,7,8

The condition that S4 equals 1 is the condition that the right hand digit of the sum is 4,5,6,7. This will be true in the cases shown in Table 10-8.

TABLE 10-8

A	*B*
0	4,5,6,7
1	3,4,5,6
2	2,3,4,5
3	1,2,3,4
4	0,1,2,3
5	0,1,2,9
6	0,1,8,9
7	0,7,8,9
8	6,7,8,9
9	5,6,7,8

The condition that S8 equals 1 is the condition that the right hand digit of the sum is 8 or 9. This will be true in the cases shown in Table 10-9.

TABLE 10-9

A	*B*
0	8,9
1	7,8
2	6,7
3	5,6
4	4,5
5	3,4
6	2,3
7	1,2
8	0,1
9	0,9

If we use the same method as last time, however, we head into difficulties. We find so much complexity that we need to vary our attack. What can we do?

Well, suppose for a moment we had only even digits to consider. Then our problem would be very much simpler. The three tables would become:

TABLE 10-10

A	*B* if S2 = 1	*B* if S4 = 1	*B* if S8 = 1
0	2,6	4,6	8
2	0,4	2,4	6
4	2,8	0,2	4
6	0,6	0,8	2
8	4,8	6,8	0

And if each of *A* and *B* were increased by 1, then the sum would be increased by 2.

We can express this as a kind of classification type circuit, as shown in Figure 10-5. The arrowhead sign ∇ in this circuit diagram represents a rectifier or diode, which permits electric current to flow only in one way, in the direction of the point of the arrowhead.

To sum up then, we can obtain the output (the sum of two single decimal digits) at the expense of the number of switch contacts shown in

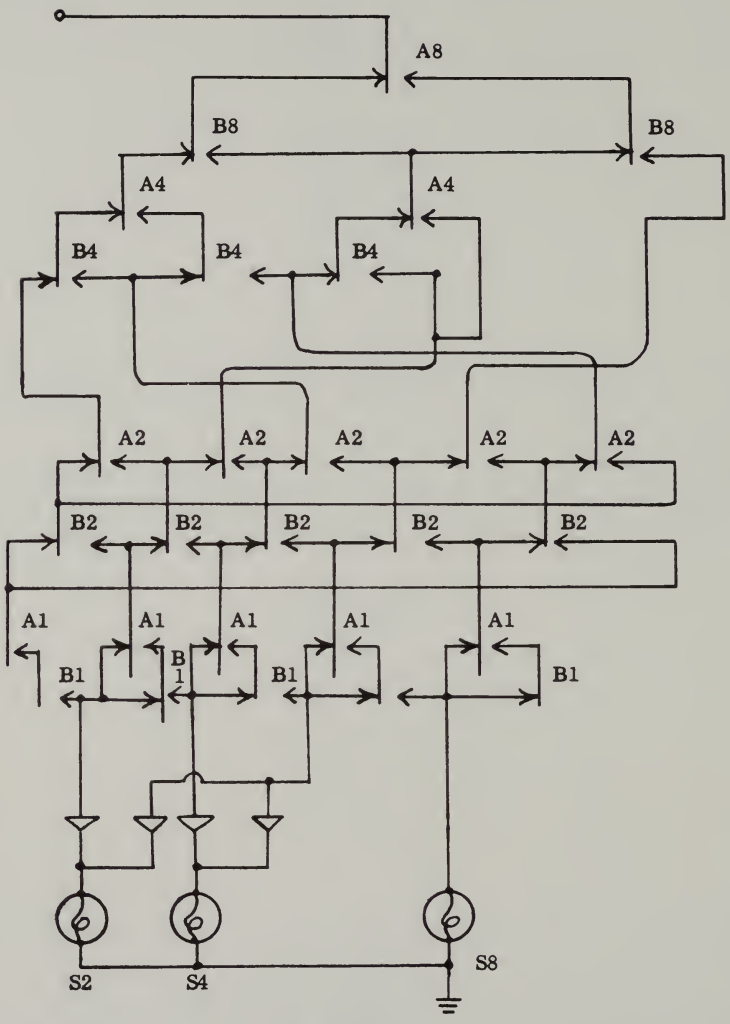

A8
B8
B8
A4
A4
B4
B4
B4
A2
A2
A2
A2
A2
B2
B2
B2
B2
B2
A1
A1
A1
A1
A1
B1
B1
B1
B1
S2
S4
S8

Figure 10-5

Table 10-11, a total of 44:

TABLE 10-11

Output	No. of Switch Contacts Required
S1	2
C1	14
S2, S4, S8	28
	44

The number of contacts, 44, could probably be reduced to a slightly smaller number by considerable further effort on the logical design of decimal addition table circuits. But even so, 44 is considerably less than the 121 switch contacts required in the first circuit (Figure 10-1).

Chapter 11

LARGE MACHINES THAT COMPUTE: THE PROBLEMS OF STORAGE, TRANSFER, AND ORGANIZATION

1. Other Problems in Computing Circuits

There are of course a great number of other problems in circuits for computing machines besides the addition of two single decimal digits. Let us take a look at some of these other problems. And to make the examples simpler and the principles more evident, let us stop dealing with decimal digits, and deal with pure binary digits instead.

The addition table for the binary number system is:

+	0	1
0	0	1
1	1	10

The multiplication table for the binary number system is:

×	0	1
0	0	0
1	0	1

Because of the great simplicity of these tables, it is not surprising that many automatic computers have been designed to operate in the pure binary number system.

Now the first problems which we shall consider are the basic problems in any automatic digital computer: how shall we store numbers? how shall we perform a sequence of calculations automatically?

2. The Organization of an Automatic Computer

A good mental picture of the working of an automatic computer is an isolated telegraph system, possessing a number of communicating stations and a traffic control. The messages that this telegraph system handles are usually items of information of standard length, with a standard number of digits.

One of the stations is called INPUT. Here information comes from the outside world into the telegraph system; at the input station, the information is put into a form ready to be sent somewhere else in the system. The form of input, may, for example, be punched paper tape; and the punches in the tape convey certain information which is accepted by the paper tape reader and converted into electrical signals received by the telegraph system.

Another of the stations is called OUTPUT. Here information that the telegraph system has produced is given back to the outside world. The form of output may, for example, be an electric typewriter, which is impulsed by the signals in the telegraph system to type out certain digits.

There exist in the system a large group of stations called STORAGE NO. 1, STORAGE NO. 2, STORAGE NO. 3, and so on. In these stations information may be stored without undergoing any change, while waiting for some other part of the system to call for the information at a suitable time and do something more with it.

A very important station in the system with platforms for several incoming pieces of information is the CALCULATOR or ARITHMETIC UNIT. This station may be thought of as combined with a factory, a calculating circuit that can accept several pieces of information and manufacture new pieces of information out of them.

For example, the Arithmetic Unit or calculating circuits may have four platforms. On two platforms, the circuit may take in two numbers, such as 1101 and 111. On the third platform, the circuit may take in an operation order to subtract (or multiply, or find which of the two numbers is bigger, etc.). On the fourth platform the circuit delivers the result; in the case of "subtract", 110, the result of combining 1101 and 111 according to the order "subtract the second number from the first one".

Connecting all these stations is the main telegraph line or BUS, consisting of one wire for each binary digit simultaneously dispatched through the system. If an automatic computer transfers at any one time ten decimal digits each expressed as four binary digits, then the bus will have forty wires or lines, a *forty-line bus*.

To perform useful work with this telegraph system, we must have some ways of organizing traffic through it. This is the duty of the CONTROL UNIT. The most automatic way of sending information through the system is:

(1) At any one time connect just two telegraph stations to the bus, such as Chicago and Denver.

(2) Specify the direction of traffic, such as "from Denver to Chicago".

Then as soon as the proper connections have been completed, send the signal "go", and the information at Denver will be dispatched automatically to Chicago.

Now, just exactly how do these operations take place?

3. A Concrete Example of Organization

Suppose we draw a very simple example of the kind of a telegraph system we are discussing.

Looking at Figure 11-1, we notice the names of four stations, called Atlanta, Boston, Chicago, Denver, Atlanta is represented by two relays, S1(2) and S1(1), standing for Storage Register NO. 1. S1(2) contains the second binary digit or twos digit; S1(1) contains the first binary digit or units digit. Boston is Storage Register No. 2, with two relays S2(2) and S2(1). Chicago is Storage Register No. 3, with two relays, S3(2) and S3(1); and Denver is Storage Register No. 4, with two relays S4(2) and S4(1). The connecting telegraph trunk or bus consists of two lines, one for each of the two binary digits, a twos line and a ones line. The information that can be stored in any of these four registers is 00, 01, 10, and 11, depending on whether the twos relay or the ones relay is energized (storing a 1) or not energized (storing a 0). The four registers themselves evidently have codes too, 00, 01, 10, and 11, which are expressed in the positioning of the Select Sending Register Relays (the *P* relays) and the Select Receiving Register Relays (the *Q* relays).

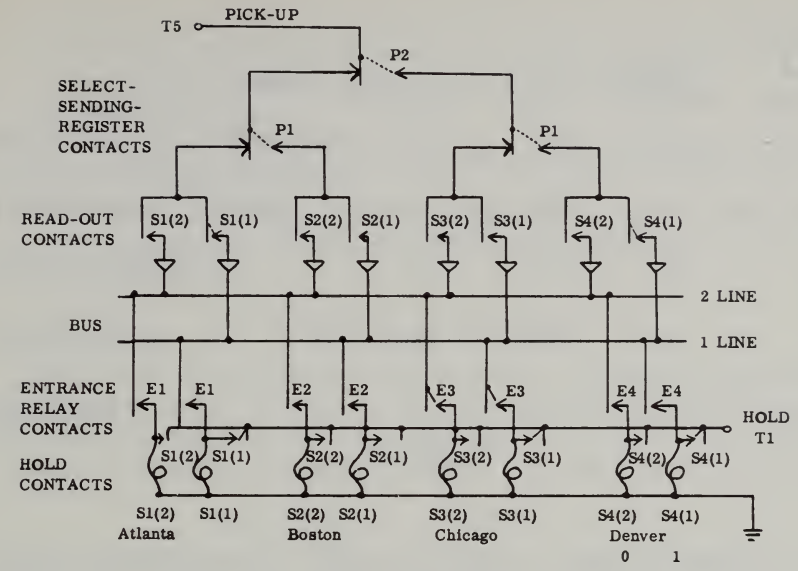
Now, how does all this work together?

Suppose that Denver stores the information 01, and we wish to transfer this information to Chicago. This means that Storage Register No. 4 stores 01—that relay S4(2) is not energized (contains a 0), and that relay S4(1) is energized (contains a 1). Consequently, the hold contact for S(4)1 is closed, and the read-out contact for S4(1) is closed, but for S(4)2 containing 0 both these contacts are open.

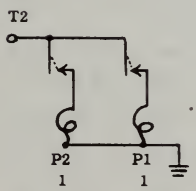
The code for the sending register S4 is 11; consequently relays P2 and P1 (Diagram b) are energized, and their contacts (the Select Sending Register Relay contacts in Diagram (a)) are closed.

The code for the receiving register S3 is 10, and, consequently, in the Select Receiving Register Relays relay Q2 is energized and relay Q1 is not energized. Because of the circuit in Diagram (d), only the relay E3 is energized, closing the two E3 Entrance Relay contacts in Diagram (a).

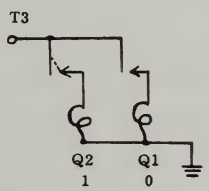
Now we can see in Diagram (a) a path from the source of current T 5, through the Select-Sending-Register contact network, through the S4 read-out contacts, into the bus, through the E3 Entrance Relay contacts, into the pick-up coils of the relays for S3, so that S3(1) is picked up,



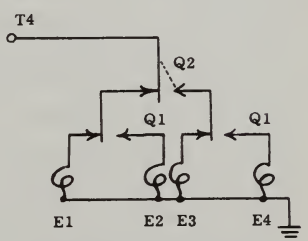
STORAGE RELAYS
Diagram (a)



SELECT-SENDING-REGISTER RELAYS
Diagram (b)



SELECT-RECEIVING-REGISTER RELAYS
Diagram (c)



ENTRANCE RELAYS
Diagram (d)

Figure 11-1

energized, but S3(2) is not energized. In this way the information 01 is transferred from Storage Register S4 into Storage Register S3.

4. Timing

Now clearly we need to be careful about the timing of these circuits. There are five terminals T1, T2, T3, T4, and T5, which provide current to the relays according to a pattern of timing, which is shown in Figure 11-2.

Relay Contacts Read Through	Relays Energized	Circuit	Terminal
S	S	(a)	T1
—	P	(b)	T2
—	Q	(c)	T3
Q	E	(d)	T4
P, S, E	S	(a)	T5

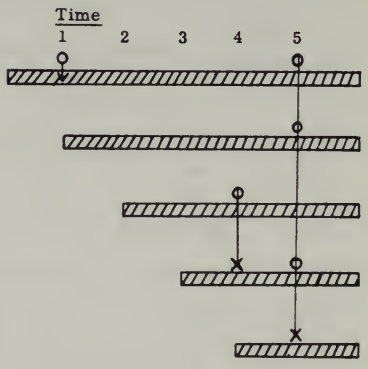


Figure 11-2

Let us consider time 1. At this time, the current running from T1 to ground passes through both any closed Hold contacts and the coils of certain Storage Register (S) relays—in particular, S4(1), since that Storage Register 4 by a previous operation contains 01; S4(1) is now energized by continuous current from terminal T1 to ground.

Let us pass to time 2, and look for terminal T2. At time 2 we see that the SELECT-SENDING-REGISTER relays (P) are energized according to a pattern (11) expressed by unnamed contacts. This pattern is established in a previous operation by the program of the computer. The program calls for selecting the sending register corresponding to code 11.

Now let us pass to time 3, and look for Terminal T3. We see that the SELECT-RECEIVING-REGISTER relays Q2 and Q1 are energized according to the pattern 10. This pattern is also expressed by unnamed contacts, which by the program of the computer at this time call for code 10, corresponding to Storage Register 3.

Passing to time 4, we look for terminal T4, and find that by the positioning of the Q contacts in the circuit of Diagram (d), entrance relay E3, and only E3, is energized. We have thus connected the coils of register S3 through the E3 contacts to the bus, and therefore only S3 can receive a signal from the bus.

5. Transferring Information

We have now completed all the preparations needed to transfer information from register S4 to register S3. We now pass to time 5. Pulsing terminal T5, we see that the pulse of current flows as follows:

(1) through the selection circuit that selects the sending register S4;
(2) through the read-out contacts of the sending register S4;
(3) through the rectifiers (which prevent back circuits);
(4) through the bus;
(5) through the contacts of the entrance relay belonging to the receiving register S3;
(6) into the coils of the receiving register S3 (naturally and properly only the ones relay of S3 is energized, however)
(7) to ground.

6. The Scheme is General

This then is an illustration of the principles for transferring information from one register to another. The scheme is entirely general; a pattern of information "written" in one register is "copied" into another.

And the repetition of instruction after instruction, each of the form

"From register .... to register ----"

where the two blanks are filled in with different register names is sufficient (with a few artifices, such as those that take care of changing from one program to another) to carry out a program of automatic computation of almost unlimited complexity.

Chapter 12

LARGE MACHINES THAT COMPUTE: ADDITION, SUBTRACTION, MULTIPLICATION, AND DIVISION OF BINARY NUMBERS

1. Addition of Two Numbers of Four Binary Digits Each

Other problems which occur in the design of large machines that compute are (1) dealing with numbers of more than one digit, and (2) carrying from one column to the next column. To illustrate these problems and their solutions, let us consider circuits which will add two binary numbers of four digits each.

Adding in binary notation is defined by the binary addition table:

+	0	1
0	0	1
1	1	10

For example, let us add binary 1101 (which is eight plus four plus one, or thirteen) and binary 1011 (which is eight plus two plus one, which is eleven):

$$\begin{array}{r} 1101 \\ +\ 1011 \\ \hline 11000 \end{array}$$

The result is 11,000 (which is sixteen plus eight, or twenty-four). We may obtain this answer by the following "schoolboy" routine: (column 1, starting from the right) 1 and 1 is 10, put down the 0, and carry 1; (column 2) 0 and 1 is 1, and 1 to carry, is 10, put down the 0 and carry 1; (column 3) 1 and 0 is 1, and 1 to carry is 10, put down 0, and carry 1; (column 4) 1 and 1 is 10, and 1 to carry is 11, put down 1 and carry 1; (column 5) put down the 1 carried.

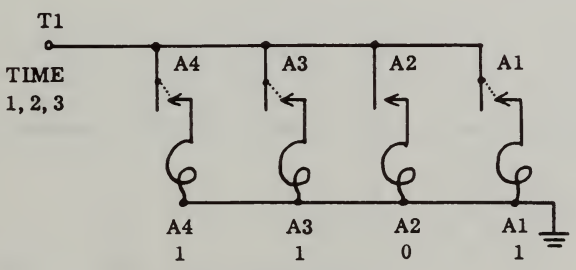
2. Boolean Algebra Equations for Binary Addition

With this example in front of us, let us consider what will be the Boolean algebra equations, by means of which the addition will be ac-

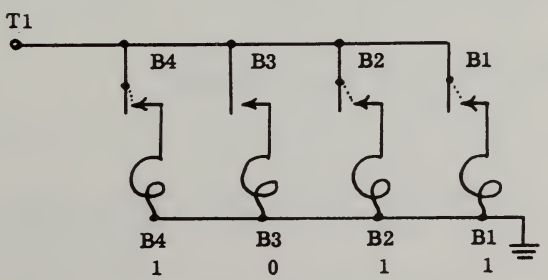
complished, for these equations will be a direct guide to the circuits for accomplishing the addition.

First, let A and B be the two binary numbers of four digits to be added. Let A4, A3, A2, and A1 be the four digits from left to right of the first number, which we may call the "Augend" (the number "to be increased"); and let B4, B3, B2, and B1 be the four digits of the second number to be added, which we may call the "Addend" (the number "to be added"). These two numbers will be stored each in a register consisting of four relays. If a relay is energized, it holds a binary digit one; if a relay is not energized it holds a binary digit zero. At some previous time these relays have been energized or not energized; and now as we start examining the circuits, Terminal T1, through the hold contacts of the relays, holds them energized or not energized during all the time that the later circuits operate. (See Figure 12-1). Let the sum of these two numbers be S; let S_i or Si be the ith digit of S counting from right to left. Let C_i or Ci be the carry digit to be added in column i resulting from the addition in column $i - 1$.

Now how do we design a circuit which will give us the sum S, which will be expressed in five relays (some of the time, four relays) S5 to S1?



AUGEND REGISTER



ADDEND REGISTER

Figure 12-1

The conditions for S_i and C_i may be expressed completely in Table 12-1 (except for the case of the first column of the two binary numbers):

TABLE 12-1

C_i	A_i	B_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The exception for the first column (i equals 1) is stated in Table 12-2.

TABLE 12-2

A1	B1	C2	S1
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

From these tables we can write down the Boolean equations:

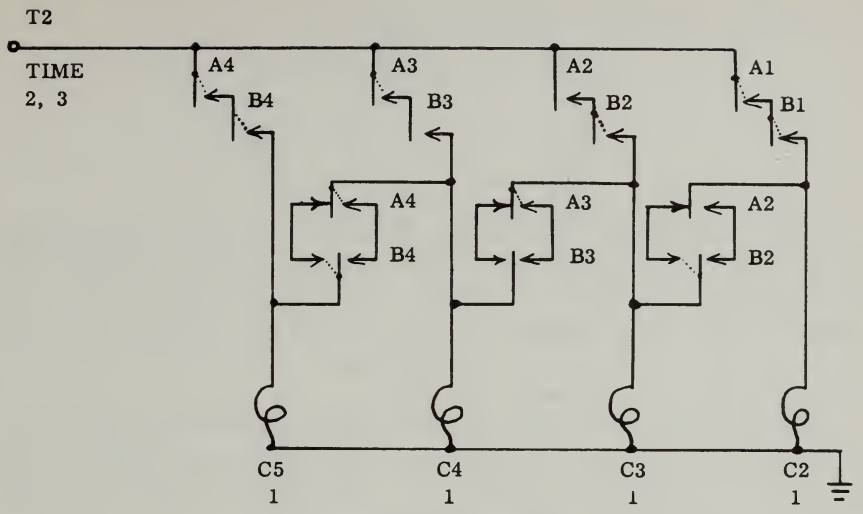
$$
\begin{aligned}
C_2 &= A_1 \cdot B_1 \\
C_{i+1} &= A_i B_i \vee C_i(A_i \cdot B_i' \vee A_i' \cdot B_i) \\
S_1 &= A_1 \cdot B_1' \vee A_1' \cdot B_1 \\
S_i &= C_i'(A_i \cdot B_i' \vee A_i' \cdot B_i) \vee C_i(A_i \cdot B_i \vee A_i' \cdot B_i')
\end{aligned}
$$

The carry digit can be expressed in terms of earlier and still earlier digits, thus:

$$
\begin{aligned}
C_{i+1} &= A_i \cdot B_i \vee (A_i \cdot B_i' \vee A_i' \cdot B_i)C_i \\
&= A_i \cdot B_i \vee (A_i \cdot B_i' \vee A_i' \cdot B_i) \cdot \\
&\quad [A_{i-1} \cdot B_{i-1} \vee (A_{i-1} \cdot B_{i-1}' \vee A'_{i-1} \cdot B_{i-1})C_{i-1}] \\
&= A_i \cdot B_i \vee (A_i B_i' \vee A_i' \cdot B_i) \cdot \\
&\quad [A_{i-1} \cdot B_{i-1} \vee (A_{i-1} \cdot B_{i-1}' \vee A'_{i-1} \cdot B_{i-1}) \cdot \\
&\quad [A_{i-2} \cdot B_{i-2} \vee (A_{i-2} \cdot B_{i-2}' \vee A'_{i-2} \cdot B_{i-2})C_{i-2}]] \\
&\quad \cdots\cdots
\end{aligned}
$$

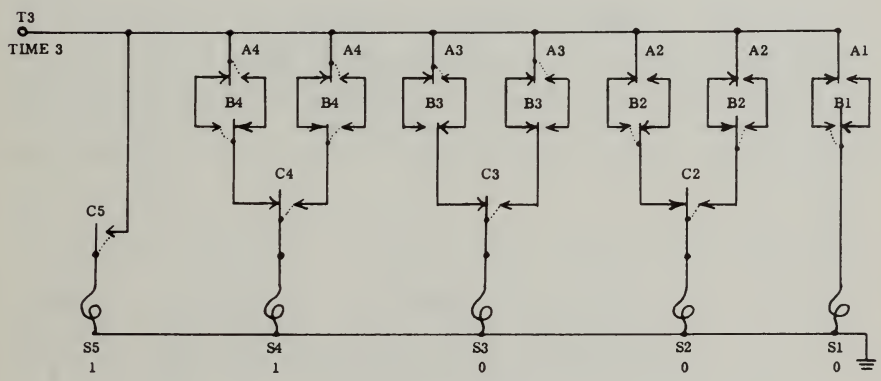
3. Circuits for Binary Addition

With these equations as guides, we can write down directly and immediately the corresponding circuits. The sequence of expressions for



CARRY REGISTER

Figure 12-2



SUM REGISTER

Figure 12-3

the carry digit C_{i+1} appears in Figure 12-2. The equations for the sum digit S_i appear in Figure 12-3.

Now how do these circuits work? Looking at these circuits, we can see that the A and B relays must be energized long enough to pick up the C relays, and in addition long enough to read through the contacts of the A, B, and C relays, and pick up the S relays. Therefore the timing

chart will be something like the following:

Terminal	Time Period: 1	2	3
T 1	√	√	√
T 2	–	√	√
T 3	–	–	√

At Time 1, Terminal T1 is energized, and the A relays in the Augend Register and the B relays in the Addend Register contain the pattern of information expressing the two numbers A and B to be added. The contacts of those relays that are energized are drawn dotted in the energized position, to represent the transferring of these contacts from the normally closed to the normal open position.

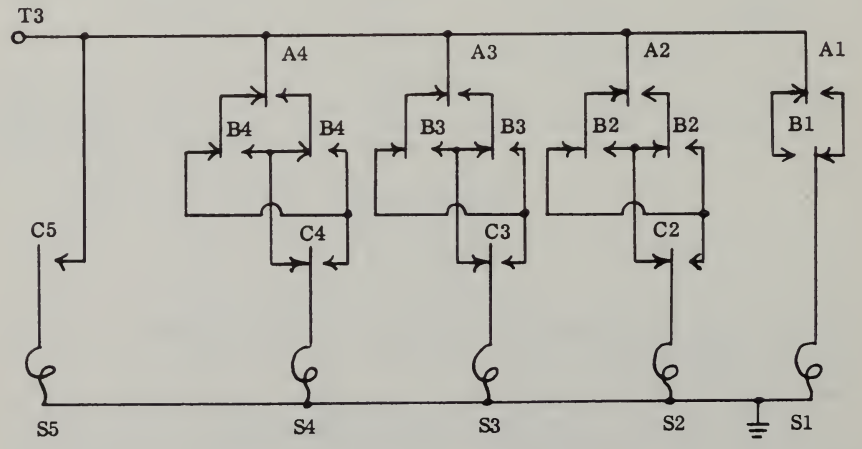
At Time 2, Terminal T2 becomes energized. The positioning of the A and B contacts in the Carry Register circuit shows that there exists a path so that all four of the carry relays are energized, for this particular choice of numbers to be added.

At Time 3, Terminal T3 becomes energized, and current flows through the contacts of the A, B, and C relays, so that the S relays are energized in the pattern 1, 1, 0, 0, 0 just as they should be.

4. Improvement of the Circuits

Now, can we improve these circuits, and if so, how?

The first improvement that we can make is in the circuit of Figure 12-3, where we can save half of the contacts for A2, A3, and A4 by the change shown in Figure 12-4. Here is an example of a rather frequent



ANOTHER SUM REGISTER CIRCUIT

Figure 12-4

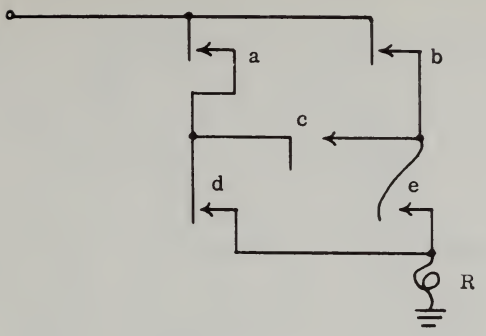


Figure 12-5

occurrence in Boolean algebra: A Boolean algebra expression can be further simplified by making use of the special properties of two-position circuit elements, even when it is difficult or impossible to simplify the expression using the ordinary, natural symbolic forms of algebra. A second example is shown in Figure 12-5: here the Boolean expression is $R = ad \vee ace \vee bcd \vee be$; and it is not at all evident from the algebra that R can be expressed as a circuit with just one switch contact for each variable.

A second improvement that we can make, if we desire to save time, is to replace C_i and C_i' by circuits which make use of the A and B contacts. We can calculate C_2' as follows:

$$\begin{aligned} C_2' &= (A_1 \cdot B_1)' \\ &= A_1' \vee B_1' \end{aligned}$$

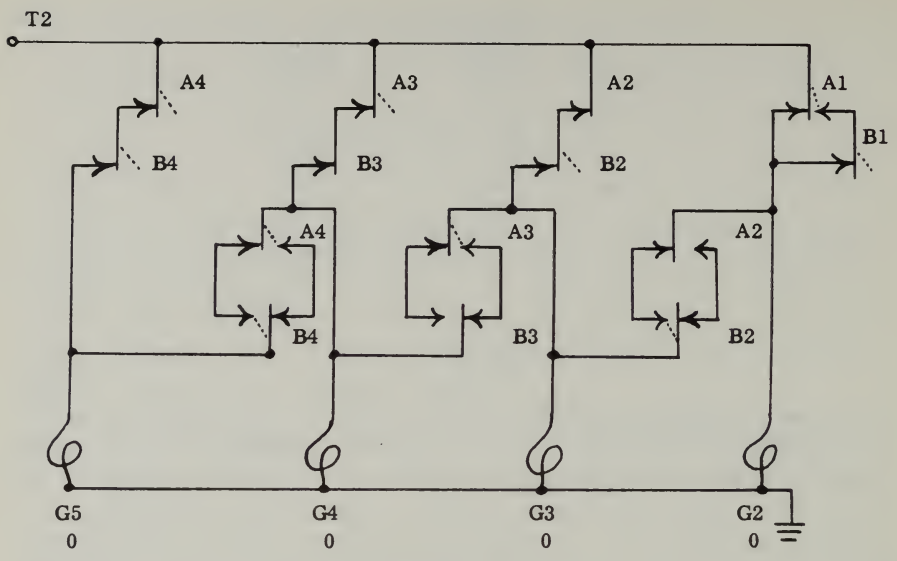
We can calculate C_{i+1}' as follows:

$$\begin{aligned} C_{i+1}' &= [A_i \cdot B_i \vee C_i(A_i \cdot B_i' \vee A_i' \cdot B_i]' \\ &= [A_i \cdot B_i]' \cdot [C_i(A_i \cdot B_i' \vee A_i' \cdot B_i)]' \\ &= (A_i' \vee B_i') \cdot [C_i' \vee (A_i \cdot B_i' \vee A_i' \cdot B_i)'] \\ &= (A_i' \vee B_i') \cdot [C_i' \vee (A_i \cdot B_i \vee A_i' \cdot B_i')] \\ &= (A_i' \cdot B_i \vee A_i' \cdot B_i' \vee A_i \cdot B_i')(C_i' \vee A_i \cdot B_i \vee A'_i \cdot B_i') \\ &= A_i' \cdot B_i' \vee C_i'(A_i' \vee B_i') \\ &= A_i' \cdot B_i' \vee C_i'(A_i' \cdot B_i \vee A_i \cdot B_i') \end{aligned}$$

Therefore we can now express C_2' and C_{i+1}' in a circuit which we may call the Carry Zero Circuit, as shown in Figure 12-6. Then the Sum Register can be expressed in terms of A and B contacts alone, as shown in Figure 12-7.

5. Subtraction of Two Numbers of Four Binary Digits Each

Suppose we wish to subtract a binary number of four digits from another binary number of four digits. But let us recall we already have an



CARRY ZERO CIRCUIT

Figure 12-6

addition circuit, and consider that we might reduce the subtraction process to the addition of a negative number. This is possible and efficient: we can make use of the *complement* of a number, and add it.

The *complement* of any number in decimal notation is found for that number by the rule: subtract each digit of that number from 9; add one to the result. Then, in order to subtract using a complement, we add the complement, and discard an extra digit one at the left. For example, suppose we desire to subtract 746 from 88965. We treat 746 as 00746; its complement is 99253 plus 1, which is 99254. Then to subtract 746, we add 99254 and discard the digit 1, at the left, thus:

88965	88965
−746	+ 99254
88219	①88219 ⟶ 88219

However in calculating machines the addition of the 1 which changes 99253 into 99254 is a nuisance; so we make use of what is called the *nines complement* (the analog of 99253) and then use what is called *end-around carry:*

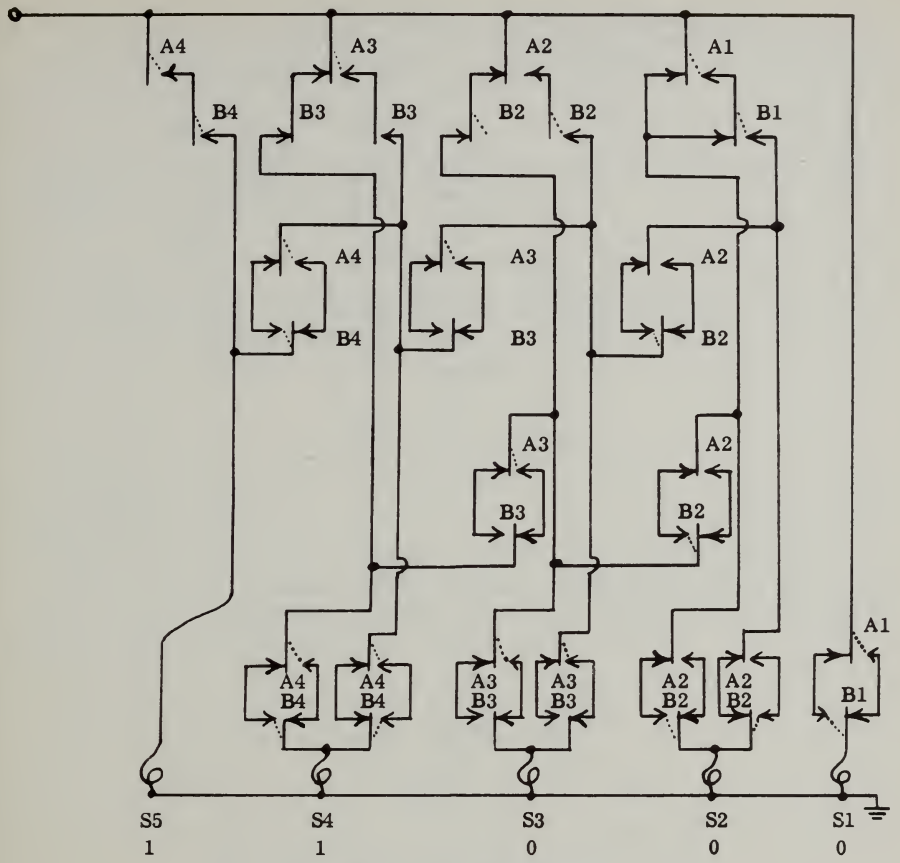
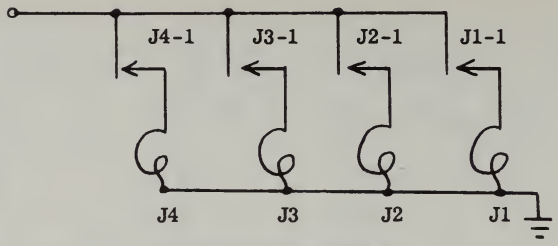


Figure 12-7. Sum Register

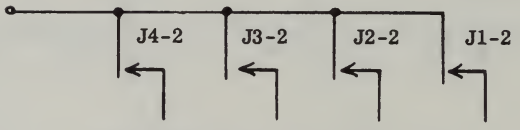
$$\begin{array}{r} 88965 \\ +\ 99253 \\ \hline (1)\,88218 \\ \quad\hookrightarrow (1) \\ \hline 88219 \end{array}$$

removing the one in the column at the extreme left and adding one in the column at the extreme right. Mathematically, it is clear that what we are doing is:

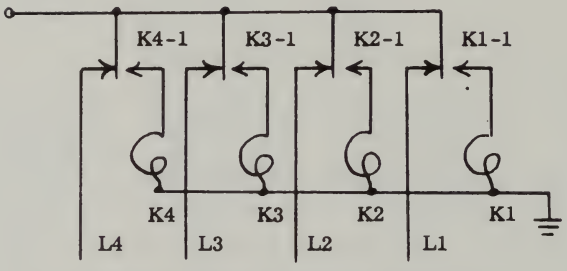
$$\begin{aligned} -746 &= -99999 + (99999 - 00746) \\ &= -99999 + 99253 \\ &= -100000 + 99253 + 1 \end{aligned}$$



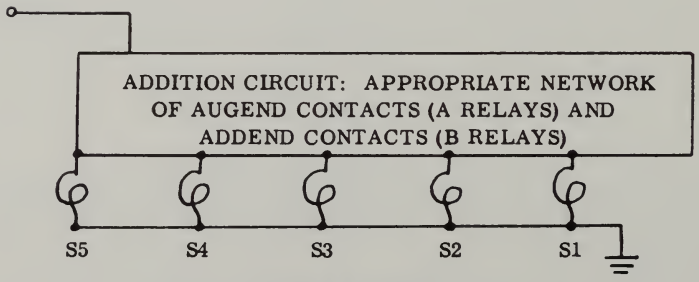
MINUEND REGISTER



MINUEND READ-OUT CONTACTS
Diagram (a)



SUBTRAHEND REGISTER AND SUBTRAHEND
ONES COMPLEMENT READ-OUT CONTACTS
Diagram (b)



SUM RELAYS
Diagram (c)

Figure 12-8

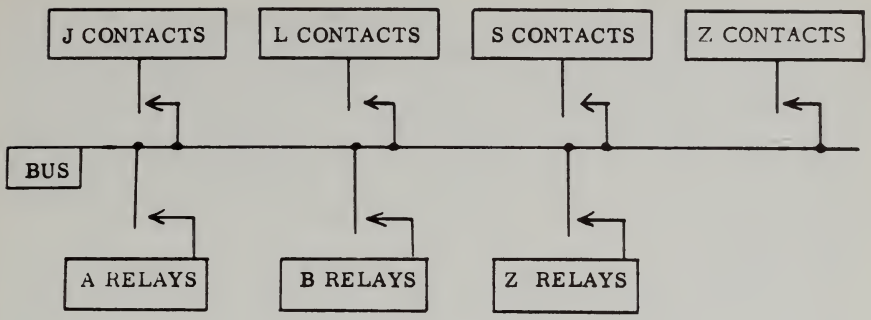


Diagram (d)

Figure 12-8 (continued)

In binary notation, the analog of the nines complement is the ones complement. The *ones complement* of a binary number is obtained by changing each digit 0 to the digit 1 and each digit 1 to the digit 0.

Let us take an example: subtract 101 from 1110. First change 101 to 0101. The ones complement of 0101 is 1010. Add the ones complement 1010 to 1110, and perform end-around-carry:

$$\begin{array}{r} 1110 \\ +\,1010 \\ \hline (1)\,1000 \\ \hookrightarrow\,1 \\ \hline = 1001 \end{array}$$

Is this correct? 1110 is 8 plus 4 plus 2, or 14; 0101 is 4 plus one, or 5; 1001 is 8 plus one, or 9. 14 minus 5 is 9. This example is therefore correct.

Therefore in order to include subtraction in our automatic computer, we can express it as the process of adding a complement.

6. Circuits for Binary Subtraction

A set of circuits which express binary subtraction is shown in Figure 12-8. Let us see exactly how they work.

The minuend (or number to be diminished) is stored in the J register, as shown in Diagram (a). The contacts J4-1, J3-1, J2-1, and J1-1 are the hold contacts keeping the relays energized to hold the information that the J register is supposed to store. When we want the information stored in this register transferred into the bus, we read out the information by means of the J4-2, J3-2, J2-2, and J1-2 contacts.

The subtrahend (or number to be subtracted) is stored in the K register as shown in Diagram (b). The normally open side of the contacts K4-1, K3-1, K2-1, and K1-1 are the hold contacts keeping the relays energized to hold the information which expresses the subtrahend. The normally closed side of these contacts provide the ones complement of the binary number stored in this register. Let us call the information provided in these four lines L4, L3, L2 and L1, and L of course is equal to the ones complement of K.

The addition circuit which we discussed in the previous section is shown schematically in Diagram (c), as some kind of a network of contacts of the Augend or A relays and the Addend or B relays, leading to the energizing of the Sum or S relays.

In Diagram (d) is a sketch of the bus (which of course actually consists of four lines and not one), and a sketch of the switching system by which the read-out contacts of various registers can be optionally connected to the bus, and the pick-up coils of various registers can also be optionally connected to the bus. Then the operation of subtraction can be accomplished by the following sequence of operations:

(1) Read-out from the J read-out contacts through the bus into the A relays.
(2) Read-out from the L wires through the bus into the B relays.
(3) Read-out from the S contacts through the bus into the Z relays, where Z is a storage register (with a special property which is explained in the next two steps) where a number can be stored while the addition circuit is reset so as to accept new information.
(4) Read out from the Z read-out contacts through the bus into the A relays -- BUT only Z4, Z3, Z2, and Z1 connect to the bus.
(5) Read-out from the Z5 read-out contact through the 1-line of the bus into the B1 relay (the end-around-carry trick).
(6) Read-out from the S read-out contacts through the bus, into wherever you wish to place the result of the subtraction.

At this point we can say to ourselves, "It is foolish to have an automatic computer that can handle only positive numbers. Our computer should handle both positive and negative numbers. How shall we arrange that?"

A good answer, though not the only one, is to agree that the extreme left hand digit of the number will tell the sign of the number, whether positive or negative. In binary notation, we can say that if the extreme left hand digit is 0, the remaining digits constitute a positive number; and if the extreme left-hand digit is 1, the remaining digits constitute a negative number.

We shall need to adjust our addition circuits for adding two numbers which are both positive or both negative or one positive and one negative. We shall also need to adjust the addition circuit to ring an alarm or perform some other kind of desired operation where the result of an addition is beyond the capacity of the computer—in the same way as an ordinary desk calculating machine rings an alarm when an operation produces a result beyond the capacity of the calculator.

A third feature of calculation for which our automatic computer will need adjustment is the location of the binary point, as for example, if we wished to add 11.01 and 111.0. But solutions to these problems are not difficult, and it is probably better to go on to the next main process, multiplication.

7. Multiplication of Two numbers of Four Binary Digits Each

In binary notation, the multiplication table becomes simply:

×	0	1
0	0	0
1	0	1

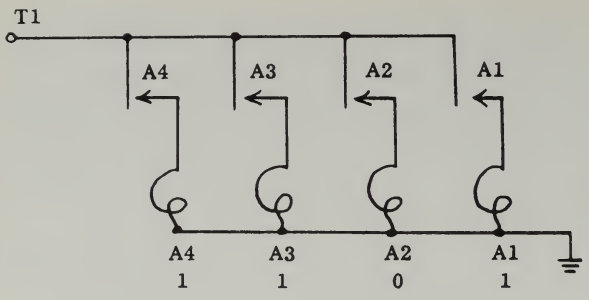
or in other words, 0 times 0 is 0, 0 times 1 is 0, and 1 times 1 is 1. In other words, multiplication reduces to either adding or not adding, and shifting. It is hard to imagine an easier multiplication table.

For example, let us multiply two binary numbers 1101 and 1011.

$$
\begin{array}{r}
1101 \\
\times\ 1011 \\
\hline
1101 \\
1101 \\
0000 \\
1101 \\
\hline
= 10001111
\end{array}
$$

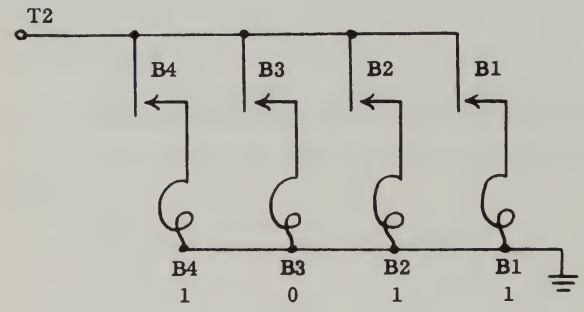
The result is 10001111, or one 128 plus no 64's plus no 32's plus no 16's plus one 8 plus one 4 plus one 2 plus one 1, or 143, which is of course what we would expect from ordinary decimal multiplication of 13(which is 1101) and eleven (which is 1011).

Figure 12-9 shows a set of circuits which will perform multiplication.



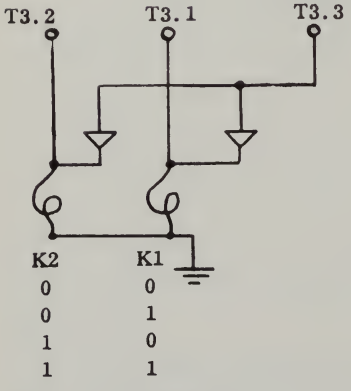
Part 1

Multiplicand Register



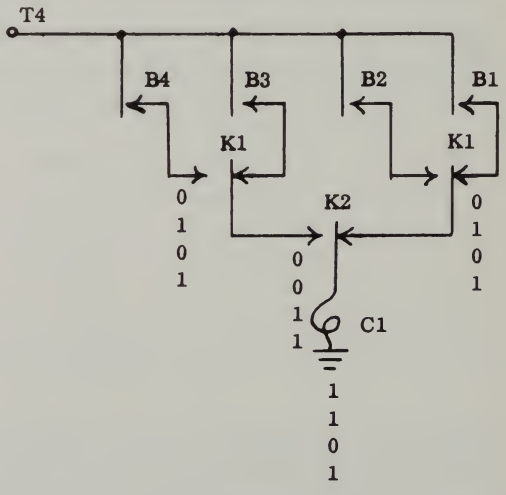
Part 2

Multiplier Register



Part 3

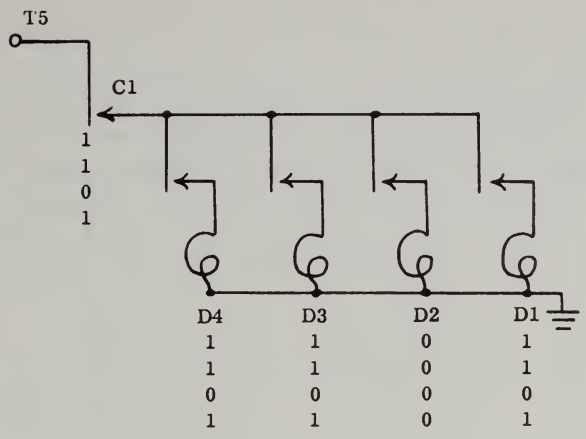
Multiplier Digit
Count Register



Part 4

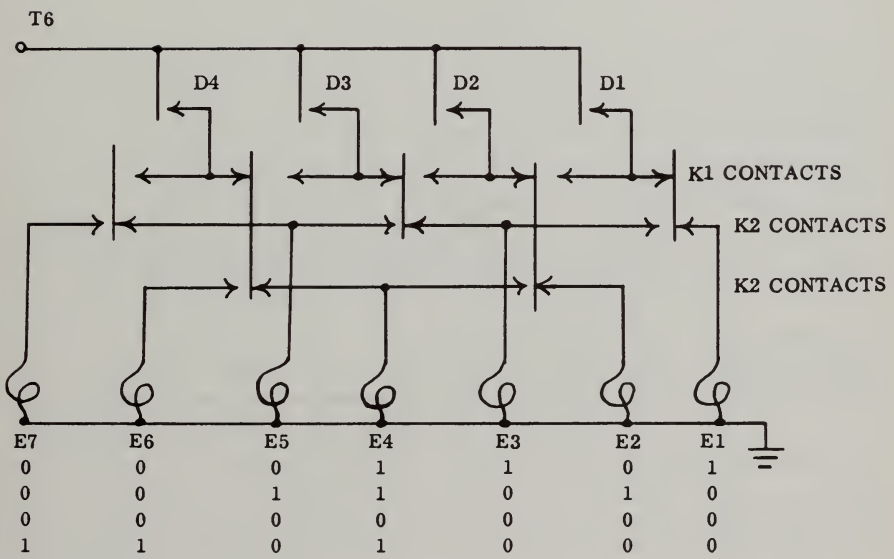
Select Multiple Register

Figure 12-9



Part 5

Multiple Register

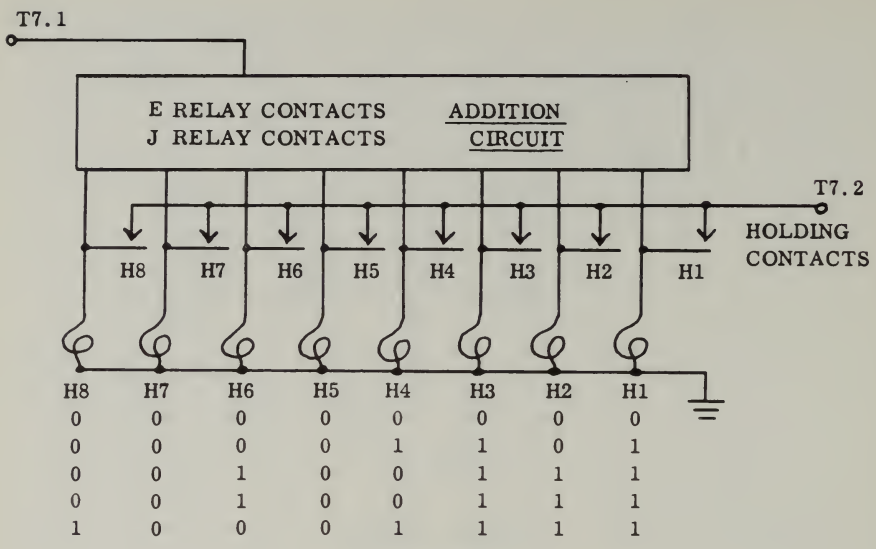


Part 6

Shift Register

Figure 12-9 (continued)

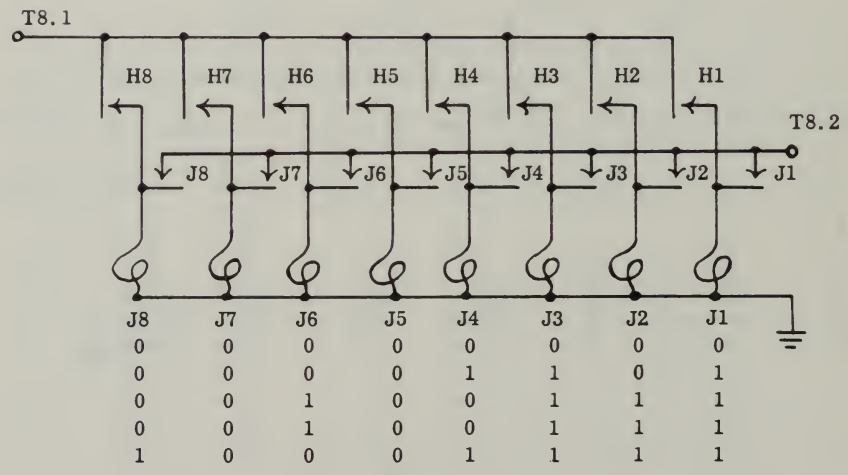
(Figure 12-9 continued on p. 132)



H8	H7	H6	H5	H4	H3	H2	H1
0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	1
0	0	1	0	0	1	1	1
0	0	1	0	0	1	1	1
1	0	0	0	1	1	1	1

Part 7

Addition Circuit



J8	J7	J6	J5	J4	J3	J2	J1
0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	1
0	0	1	0	0	1	1	1
0	0	1	0	0	1	1	1
1	0	0	0	1	1	1	1

Part 8

Partial Sum Register

Figure 12-9 (continued)

The general procedure which these circuits express is this: choose multiples of the multiplicand either 1101 or 0000 according to the successive digits of the multiplier taken from right to left 1, 1, 0, 1; shift these multiples appropriately over to the left; attend to them according to the successive positions of the multiplier digits; finally, with an addition circuit and a storage register, add the shifted multiples successively. This process is shown in terms of successive results in Table 12-3.

TABLE 12-3

Partial sum	0000 0000
1st multiple	0000 1101
New partial sum	0000 1101
2nd multiple	0001 1010
New partial sum	0010 0111
3rd multiple	0000 0000
New partial sum	0010 0111
4th multiple	0110 1000
Final sum	1000 1111

Let us now examine the parts of the circuit in detail. In Part 1 and Part 2 of the circuit we see that the multiplicand and the multiplier are stored. In Part 4 of the circuit, we succeed in making the relay C1 tell us at successive times the successive digits of the multiplier, from right to left 1, 1, 0, 1, in order that we may select the proper multiple of the multiplicand—1101 if the multiplier digit is 1, and 0000 if the multiplier digit is 0. This is made to happen because in Part 3 of the circuit the K relays are energized in the pattern 00, 01, 10, and 11.

In Part 5, the multiple of the multiplicand is selected. The successive numbers yielded by this circuit are 1101, 1101, 0000, and 1101. The contacts used are a contact of the C relay and four contacts of the A relays; the multiple is recorded in the D relays.

In Part 6, the selected multiple recorded in the D relays is shifted 0, 1, 2, or 3 spaces to the left, according to the position of the multiplier digit, again using the K relays (see Part 3). The numbers produced by the E relays are therefore:

000 1101,
001 1010,
000 0000, and
110 1000

In Part 7, the addition circuit is indicated but only by a block diagram, since it has been treated in full earlier in this chapter. The storage register, Part 8 of the circuit, is needed to transfer the result of the addition circuit back into one of its inputs. Using these two circuits the shifted multiples are added one after another as indicated in the series of successive additions shown in Table 12-3.

For this series of circuits to work properly, however, the timing of them is crucially important.

The timing of the circuits is shown in the timing chart of Fig. 12-10. A good deal of valuable information is summarized in this chart.

Successive time intervals are numbered 1, 2, 3, 4, up to 24, and are represented from left to right in the columns of the Table numbered 1 to 24. The twelve different terminals are shown in the rows of the table from top to bottom together with the relays from A to J which they energize.

Opposite each terminal, the thick horizontal line starts at the time when the terminal is energized by a source of current and stops when that terminal ceases to be energized. For example, terminal T4 is energized at times 2, 3; 7, 8; 12, 13; 17, 18; and at no other times.

Sections of different horizontal lines are shown connected by vertical lines with X's and O's. These vertical lines and marks express the functional relation of the circuits. The X designates the relays that are energized at that time; the O's designate the already positioned relay contacts through which the relays are energized. At time 8 for example, the D relays are energized by current from Terminal T5 through the A relay and C relay contacts, and also the J relays continue to be energized by current from Terminal T8.2 through the J relay hold contacts.

It should be said again that there are many ways of condensing and improving these circuits in Figure 12-9. For example, Parts 4, 5, and 6 could be combined and the C and D relays could be eliminated. But the resulting circuit would have been harder to explain and to understand than the separate circuits here shown.

8. Division of a Binary Number of Eight Digits by a Binary Number of Four Digits

It is nearly always easier to begin with an example and then go to a general rule. So, in considering long division let us as usual begin with an example, and consider the division of 1000,0101 (133 in decimal) by 1101 (13 in decimal). We perform division in binary notation in the same general way as we do in decimal notation, except that we behave as if we knew only the two digits 1 and 0.

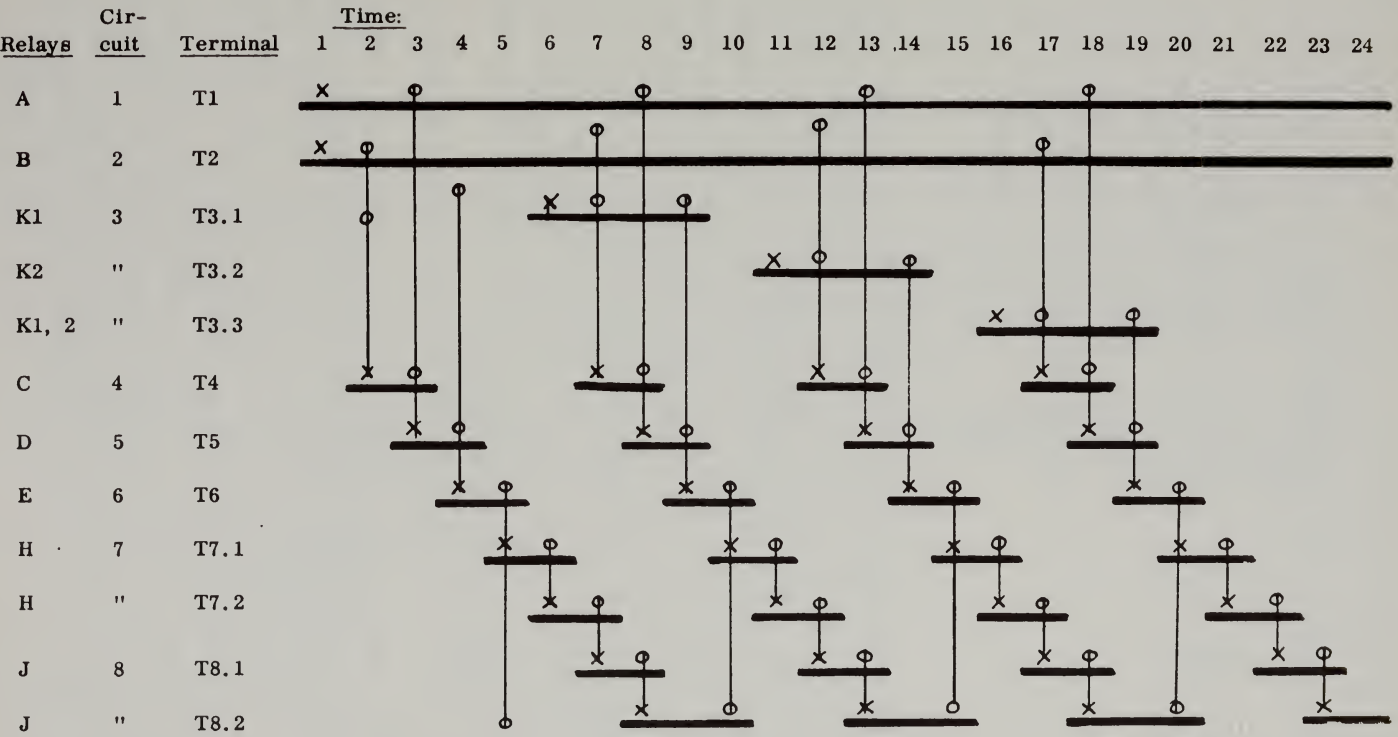

Time:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Relays
Circuit
Terminal
A 1 T1
B 2 T2
K1 3 T3.1
K2 " T3.2
K1, 2 " T3.3
C 4 T4
D 5 T5
E 6 T6
H 7 T7.1
H " T7.2
J 8 T8.1
J " T8.2

Figure 12-10

Here is the division worked out:

		01010	Quotient
Divisor:	1101	)10000101	Dividend
		0000	1st Divisor
		10000	2nd Partial Remainder
		1101	2nd Divisor
		0111	3rd Partial Remainder
		0000	3rd Divisor
		1110	4th Partial Remainder
		1101	4th Divisor
		0011	5th Partial Remainder
		0000	5th Divisor
		011	Remainder

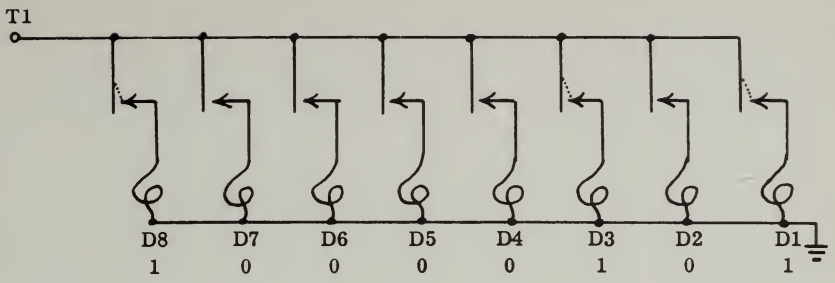
We see at once that only two multiples of the divisor are used, one times the divisor, which is the same as the divisor, and zero times the divisor, which, of course, is zero in every digit. No other multiples of the divisor are needed. If we simply compare the divisor with the partial remainder at any point in the division, we can determine whether the digit to be put in the quotient is 1 or 0. Division is reduced to subtracting or not subtracting, and shifting. What could be an easier process?

As before, in order to keep the circuits simple, let us ignore some of the finer points: fractions; the binary point (the analog in the scale of two to the decimal point in the scale of ten); positive and negative numbers; the size of the numbers, etc. For, here we are engaged in illustrating the basic underlying process of applying on-off logical reasoning to the performance of arithmetical division.

The circuit is shown in Figure 12-11. Let us examine the parts of the circuit.

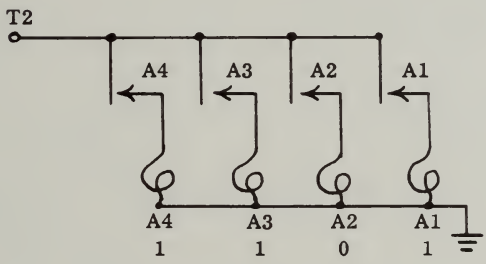
In Part 1 of the circuit, Terminal T1 is energized at the start of the time we are considering, and holds up the relays storing the dividend through their hold contacts. The actual binary number which these relays store, of course, depends on something that happened before the time at which we begin. In the same way, Part 2 of the circuit shows the relays which store the divisor, and Terminal T2 holds them up.

Now different things have to happen at different stages during the division. So we need to have some relays which will tell us in what stage we are during the process of division. This is the function of the K relays of Part 3 of the circuit. The stages that they detect and report



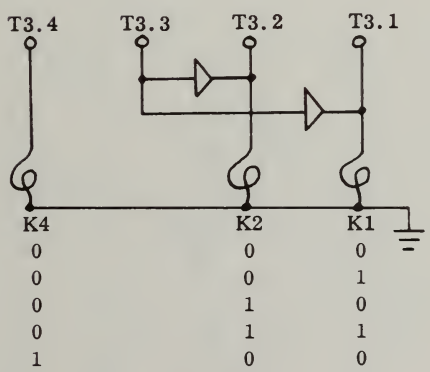
Part 1

Dividend Register



Part 2

Divisor Register



Part 3

Count-Quotient-Digit Circuit

Figure 12-11

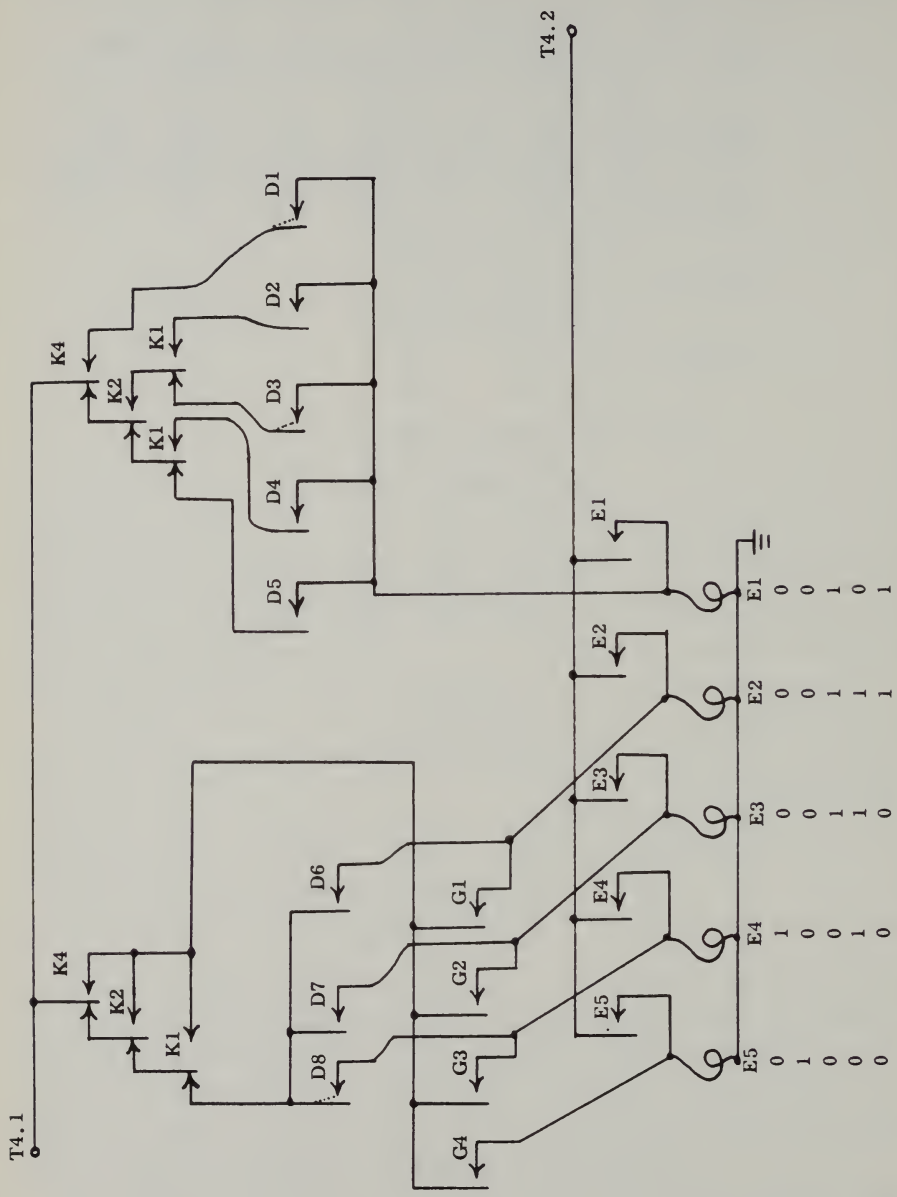
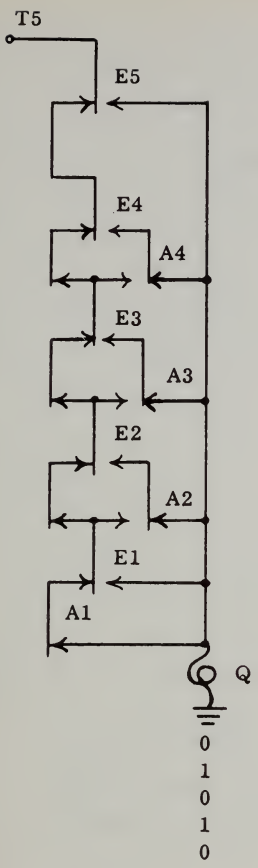


Figure 12-11 (continued)



Part 5

Determine-Quotient-Digit Circuit

Part 6

Quotient Register

Holding Contacts

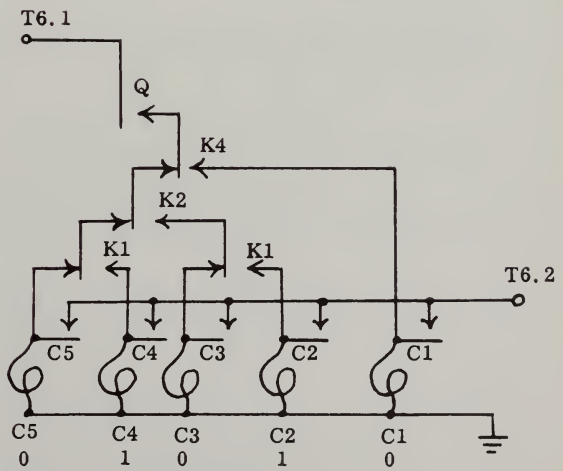
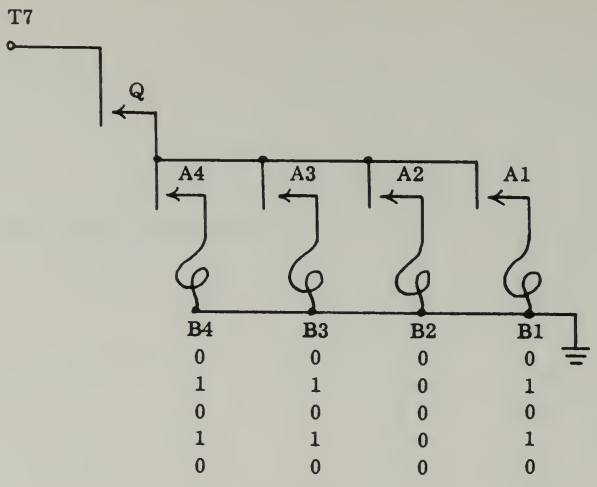
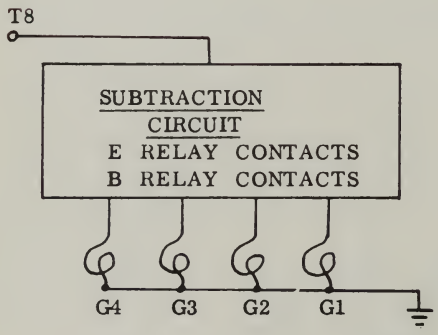


Figure 12-11 (continued)



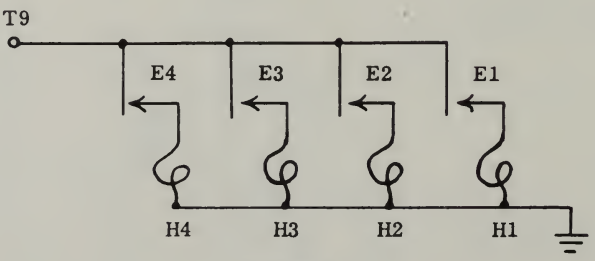
Part 7

Select-Divisor-Multiple Register



Part 8

Subtraction Circuit and Next Partial Remainder Register



Part 9

Final Remainder Register

Figure 12-11 (continued)

are stages 0, 1, 2, 3, and 4, corresponding to the first, second, third, fourth, and fifth subtraction of a divisor from the existing partial remainder. The existing partial remainder (see Part 4 of the circuit) first consists of the four left-hand digits of the dividend; but at later stages it consists of the result of a subtraction together with "bringing down" one more digit of the dividend. The circuit of Part 4 arranges that at each stage of the division we have just the partial remainder that we desire stored in the E relays. We have to look ahead to Part 8, of course, where the result of the subtraction is obtained, and for the moment take on faith that the G relays there energized will give us through their contacts in Part 4 of the circuit the result of the subtraction that we need.

The function of Part 5 of the circuit is to decide whether or not the divisor "goes" into the existing partial remainder or "doesn't go". To make this decision, we must compare the partial remainder in the E relays with the divisor in the A relays. The divisor "goes" and yields 1 as the quotient digit Q if ond only if the partial remainder is larger than or equal to the divisor. Now how shall we decide this?

We make this decision by comparing the two numbers digit by digit. The number E is greater than or equal to the number A (and the quotient digit is 1) if and only if:

E5 is greater than A5, or
E5 is equal to A5, and E4 is greater than A4, or
E5 is equal to A5, and E4 is equal to A4, and E3 is greater than A3, or
E5 to E3 is equal digit by digit to A5 to A3, and E2 is greater than A2, or
E5 to E2 is equal digit by digit to A5 to A2, and E1 is greater than or equal to A1.

But how do we put this condition into Boolean algebra?
When we deal with the Boolean algebra of 1 and 0, it is true that:

(1) $x > y$ if and only if $x = 1$ and $y = 0$, or $x \cdot y'$;
(2) $x = y$ if and only if $x = 1$ and $y = 1$, or $x = 0$ and $y = 0$, or $xy \vee x' \cdot y'$

In other words, the condition is equal to the following:

$$
\begin{aligned}
Q = \quad & E5 \cdot A5' \\
& \vee (E5 \cdot A5 \vee E5' \cdot A5') \cdot E4 \cdot A4' \\
& \vee (\quad " \quad) (E4 \cdot A4 \vee E4' \cdot A4') \cdot E3 \cdot A3' \\
& \vee (\quad " \quad) (\quad " \quad) (E3 \cdot A3 \vee E3' \cdot A3') \, E2 \cdot A2' \\
& \vee (\quad " \quad) (\quad " \quad) (\quad " \quad) (E2 \cdot A2 \vee E2' \cdot A2') \, (E1 \cdot A1' \vee E1 \cdot \\
& A1 \vee E1' \cdot A1'
\end{aligned}
$$

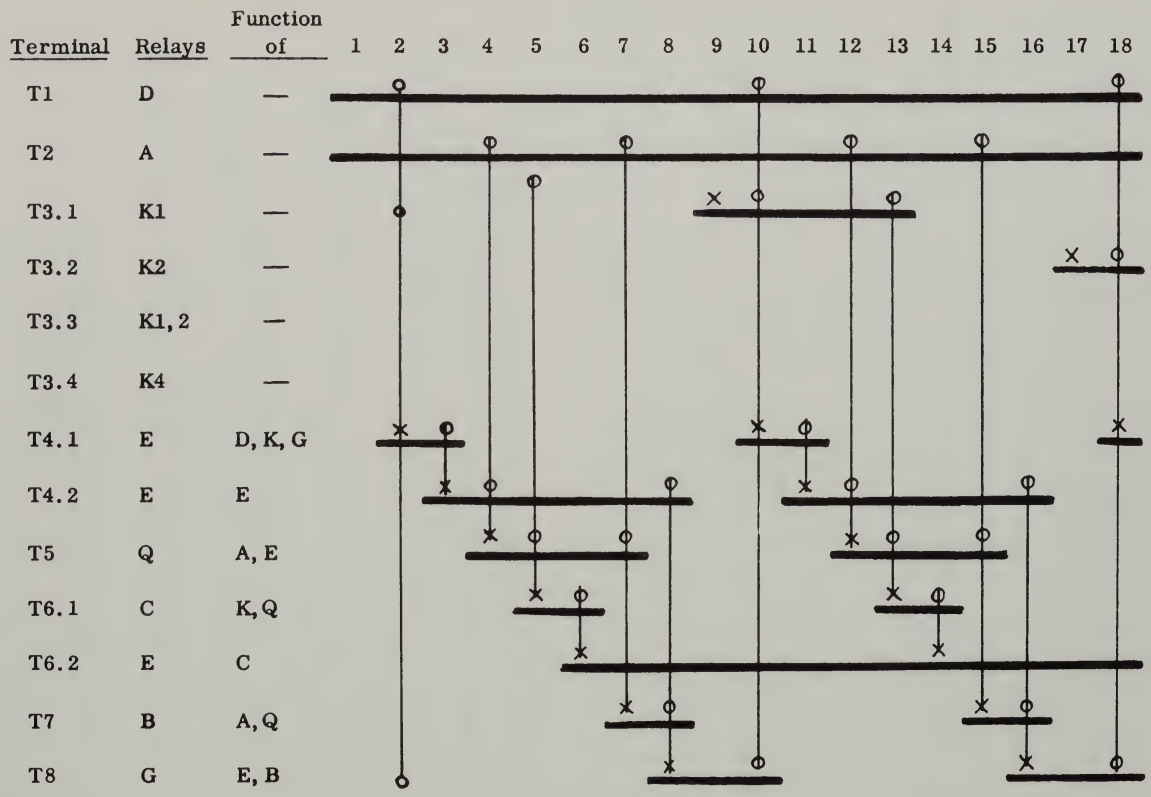


Figure 12-12. Timing Chart for Division

But we know that A5 is 0, and hence A5′ is 1, and so the circuit becomes what is shown in Part 5 of Figure 12-11.

As soon as we have found each quotient digit, we need to store it, so that we shall know the whole quotient when we finish the division. This duty is performed by the circuit of Part 6, which routes the quotient digit according to the stage when it is found, into the appropriate position in its quotient.

We now need to determine the multiple of the divisor that depends on the quotient digit. This is the function of Part 7 of the circuit, which gives the divisor itself if the quotient digit is one and zero for all digits if the quotient digit is 0.

In Part 8, the divisor multiple is subtracted from the existing partial remainder, resulting in the new partial remainder. This is indicated schematically because the actual circuits for subtraction have been discussed previously.

The timing of the division circuits, up to the end of the first two quotient digits, is shown in the timing chart of Figure 12-12. The same conventions are used there as in the timing chart for the multiplication process.

This, then, brings us to the end of the present illustration of the relation of large machines that compute to symbolic logic and Boolean algebra. The intimate relation between the on-off-ness of circuit elements, and one-zero-ness of Boolean algebra is clear. Although this illustration has all been in terms of relays, nevertheless there is no restriction to relays. Electronic tubes, transistors, magnetic cores, germanium diodes, and many other types of on-off circuit elements may be subjected to effective analysis and synthesis by means of Boolean algebra, as soon as OR, AND, and NOT have been represented in circuit elements.

Chapter 13

THE ALGEBRA OF STATES AND EVENTS: THE BASIC IDEAS

1. Boolean Algebra Extended to Include Time

In the foregoing chapters we have described Boolean algebra and shown many of its applications—to classes, statements, conditions, truth values, switches, relays, and other types of elements. Many of the meanings of common ordinary expressions of language have received symbols and become recognized as ideas that we can calculate with using Boolean algebra. These expressions include AND, OR, NOT, EXCEPT, OR ELSE, NOT BOTH, NEITHER ... NOR, IS INCLUDED IN and others.

Now there are some more common ordinary expressions of language that are used frequently in talking about the relations of statements and conditions and that are also used frequently in talking about the design of circuits using on-off circuit elements. These are expressions having to do with time, such as WHEN, BEFORE, AFTER, WHILE, DURING, HAPPEN, CHANGE, BEGIN, FINISH, and similar expressions. These expressions deal with *states* and *events*, with conditions that may change as time changes. In these expressions, we can appropriately identify ideas, give symbols to them, and calculate with them. We obtain something larger than Boolean algebra; we obtain something that might be called "Boolean calculus", for like ordinary calculus it deals with changes of functions. We obtain something larger than the "algebra of classes"; we obtain something that might be called the "algebra of states and events". This is the name which we shall preferably use, because it conveys more meaning than "Boolean calculus". This new territory is equal to Boolean algebra extended to provide for changes as time goes on.

In Boolean algebra we considered a set of binary variables p, q, r where we knew that p had to be 0 or 1, q had to be 0 or 1, r had to be 0 or 1; and we knew that p, q, and r held a single value during any one problem, even if we did not know which value it was. In the algebra of states and events, we consider the same set of binary variables p, q, r, but we consider them to be functions of time t measured or counted in

ordinary numbers. At any one time p, q, r has to be either 0 or 1 or undefined but over an interval of time the value of p, q, or r may change from 0 to 1 or from 1 to 0, and if the change is not instantaneous, during the change its value is undefined. We consider that we know the binary variable p completely if for each time in the interval we know whether it has the value 1 or the value 0 or is undefined.

The algebra of states and events, Boolean calculus, is of course a natural extension of Boolean algebra, bringing it into closer relation with the real world. For in the real world, the members of a class of objects enter the class or leave the class as time changes. For example, a horse becomes a current member of the class of horses when it is born, and ceases to be a current member of the class of horses when it dies. Also, a statement true at one time may not be true at another time. For example, a circuit element on at one time may be off at another time. It is customary in ordinary language to talk about one of these changes of state as an event, and so it is reasonable to speak of this extension of the algebra of classes as the algebra of states and events. The binary variable p which stood for the truth value of a statement P which did not mention time, such as "The circuit element E is on", now becomes a binary variable $p(t)$ or p_t or simply p which is equal to the truth value of a statement P which does mention time, as for example "The circuit element E is on at time t".

2. Time Units

When we start dealing with p, q, r as functions of time t, the first question we need to consider is, How finely shall we subdivide time? Ordinary differential calculus suggests that we treat time as a "continuous variable", allowing subdivision of time down to infinitely fine subdivisions, much finer than microseconds. This may be theoretically desirable, but it does not correspond very well to reality, and we are primarily interested in the practical operation of apparatus and circuits. Practically, it is useful to treat time as consisting of a succession of small finite units. For some circuits, seconds are good units, for others milliseconds, and for others microseconds. We shall suppose that we can measure time in units, and we shall often use as a unit, either a "little unit", the time that it takes a circuit element of the type we are considering to change state from 0 to 1 or from 1 to 0, or a "big unit" which is a large multiple of a little unit, such as 1000 times a little unit.

When we use "little" units of time, p will take all (or most) of a whole unit of time to go from 0 at the start of the unit of time to 1 at the end of the unit of time, and vice versa. When we use "big" units of time, p will ordinarily have just one of the values 0 or 1 for a considerable

stretch of time, and every now and then changes will take place as jumps at the beginning or end or some other point in the "big" unit of time. Of course, there will be situations in the real world where neither of these simplifying assumptions works very well, and in such cases we may need to make different assumptions.

3. Step-Functions

What will our variables p, q, r as functions of t look like when graphed? They will be "step-functions" and will look like the graphs shown in Figure 13-1. Here we have provided for 8 "big" units of time along the horizontal axis; we have assumed that it takes the circuit element we are imagining a thousandth of one of these graphed units of time to change from 0 to 1 or from 1 to 0. Also, we have assumed that changes may occur only at the junctions of the intervals.

The first graph in Figure 13-1 shows the function p, which is defined in Table 13-1.

TABLE 13-1

Continuous t	p	Counted t	p
0 to 1	0	1	0
1 to 4	1	2, 3, 4	1
4 to 6	0	5, 6	0
6 to 8	1	7, 8	1

Close to the time t equal to 4 (continuous time t), for example, we shall not know in practice whether p equals 0 or 1; and so here p is undefined. And in practice we contrive the circuit to avoid reliance on such a circuit element at such a time.

The same situation occurs when we consider real life instead of circuit elements. Suppose p is the truth value of "John Q. Smith is dead at time t". Now there is a day when John Q. Smith dies, and during that day there are moments when he is not yet dead, and moments when he is dead, and a group of moments in between when no one, friend, doctor, or scientist, can actually determine whether he is alive or dead. In those situations, p is in actual fact not defined, and there is little sense in being arbitrary about it and insisting on one definition or another; in actual fact we avoid relying on the information p at a time when it cannot be determined whether p is 1 or 0.

In Figure 13-1, we have also shown the graph of a function q, which is defined in Table 13-2.

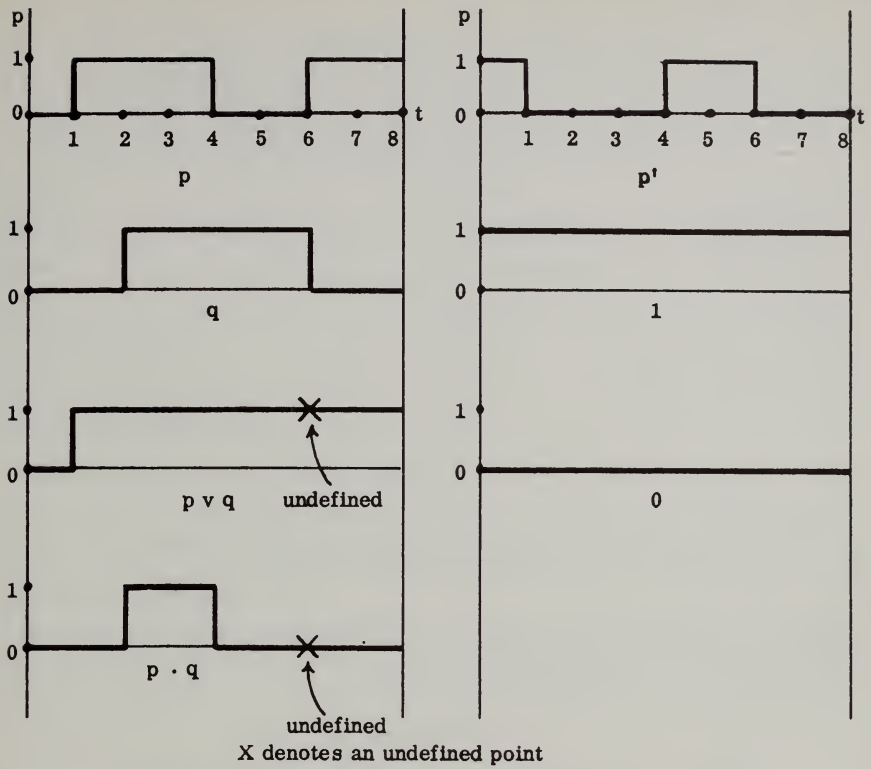


Figure 13-1. Binary Variables that are Functions of Time

TABLE 13-2

Continuous t	q	Counted t	q
0 to 2	0	1, 2	0
2 to 6	1	3, 4, 5, 6	1
6 to 8	0	7, 8	0

In Figure 13-1 are also shown the functions p OR $q = p \vee q$, p AND $q = p \cdot q$, NOT-$p = p'$, the function "always 1", and the function "always 0". Note that although $p \vee q$ and $p \cdot q$ appear to be continuous at $t = 6$, they are not, but are actually undefined at those points.

There is a second convenient way of graphing binary variables: drawing a line when the variable is equal to 1; not drawing any line at all when the variable is equal to 0. Often this style is more convenient than the first style. Figure 13-2 illustrates the same seven functions drawn again in this style. In a case where a finite length of time is

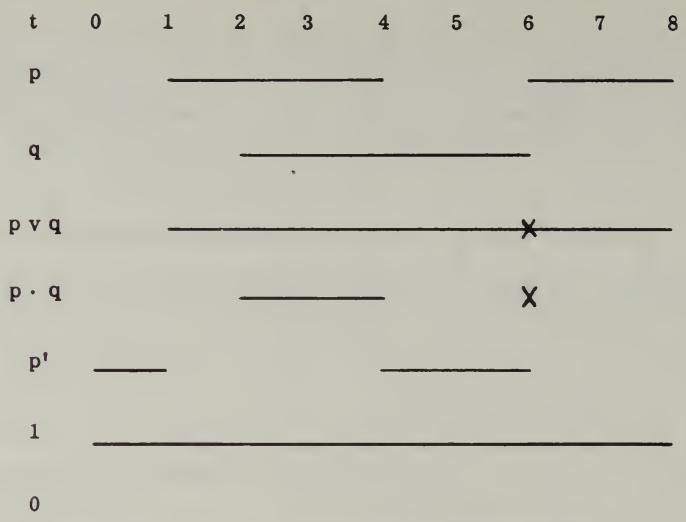


Figure 13-2. Binary Variables as a Function of Time—Second Way of Graphing Them

counted in "little" units, a table of the values of a binary variable p may look like the function in Table 13-3.

TABLE 13-3

t	p	t	p
1	0	6	1 to 0
2	0	7	0
3	0 to 1	8	0 to 1
4	1	9	1 to 0
5	1	10	0

This might express the record of a patient with health statements as expressed in Table 13-4.

TABLE 13-4

t	Statement P	t	Statement P
1	Well	6	Penicillin shot—temperature went to normal
2	Well	7	Normal temperature
3	Fell sick	8	Temperature started up again
4	Fever	9	Another penicillin shot, and down to normal
5	Fever	10	Normal temperature

4. New Operators and Their Definitions

Up to this point we still have what is almost Boolean algebra; but now we introduce new operators which express most of the ideas contained in the following words and the various synonyms for them:

WHILE	DELAY	START
DURING	ADVANCE	FINISH
WHEN	EXCEPT WHEN	END
AT	EXCEPT DURING	ENTER
IN THE EVENT OF	BEFORE	LEAVE
HAS HAPPENED	AFTER	BECOME
FOR THE FIRST TIME	BEGIN	CHANGE
FOR THE SECOND TIME	EVENT	STATE

The definitions of the new operators are in many ways rather evident and obvious; but after they are defined, they actually do give us some new powers in the application of symbolic logic and mathematics to calculations in the design of intelligent machines, and so they have some interest and importance.

The symbols for the new operators are often the initial letters of words expressing the ideas which these operators stand for. This may not be the best possible choice, but it is a reasonable and memory-aiding choice.

5. The DELAY Operator

Perhaps the most fundamental of the new operators needed for the algebra of states and events in DELAY. For example, consider the statement P:

P: Circuit element E is on at time t

It will have a truth value p or $p(t)$ or p_t which varies with time t. Associated with this circuit element E may be another circuit element F for which the following statement Q is true:

Q: Circuit element F is on at time t plus k.

where each time t in this statement Q is a time t for which statement P is true. This statement will have a truth value q or $q(t)$ or q_t which will have exactly the same time pattern as the truth value p or p_t except that it will be delayed by k units. Mathematically,

$$q(t + k) = p(t) \text{ or } q(t) = p(t - k)$$

We can of course be certain that if two circuit elements E and F actually have this kind of relation, there is a close association of cause or correlation between them.

For an example, if k equals 2, and p is the binary variable shown in the upper part of Figure 13-3, then p delayed 2 units of time is shown in the lower part of Figure 13-3. Clearly, the amount of delay k may be any number from 0 to infinity.

For designation of the DELAY operator, it is convenient to use the initial letter D:

$$D^k(p), \text{ also written } D^k p, \text{ or } p^k = p(t - k)$$

Note that in Figure 6-3 the function p has not been defined for t less than 0. Then $D^2 p$ is not defined for t less than 2. The dotted line in the diagram indicates a part of the graph of p delayed 2 units where this function has not been defined.

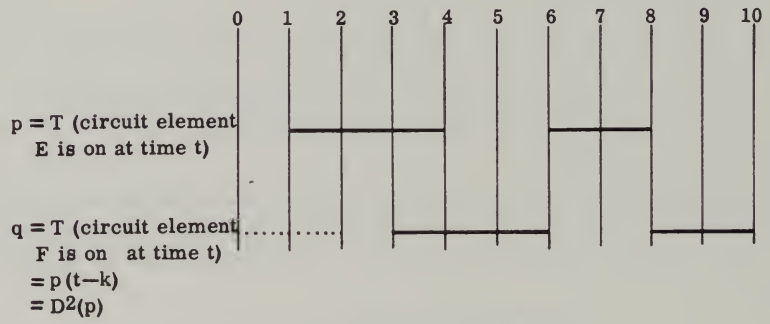


Figure 13-3. The DELAY Operator

If the delay is negative, we obtain the operator ADVANCE. If k is a positive number, q_t equals $D^{-k}(p)$, is equal to $p(t + k)$. An illustration of p advanced 3½ units appears in Figure 13-4.

One illustration where delay occurs and is important is in the operation of the hold contact of a relay, A small amount of time is needed for a

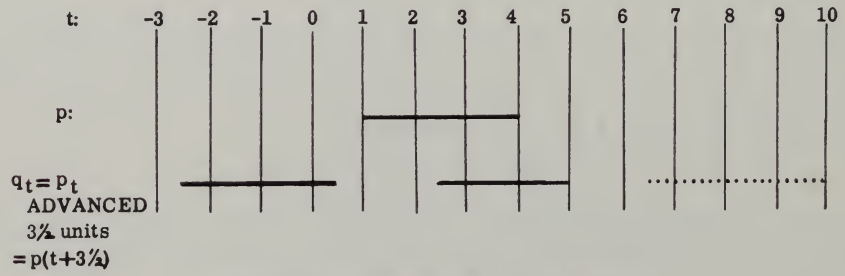


Figure 13-4. The ADVANCE Operator

relay to become energized; its contacts therefore do not close at just the same time as the coil is energized. A very fast relay is energized in 2 or 3 milliseconds; an ordinary relay, in 15 or 20 milliseconds; a slow or a *time-delay relay,* in 50 milliseconds up to 5 seconds and longer.

Consider for example Figure 13-5, which shows a relay *R,* and a terminal *T* marked "pick-up; *T* when connected to a source of current

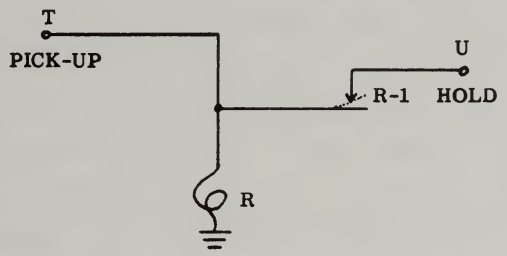


Figure 13-5. Holding A Relay Energized.

will pick up or energize relay *R.* As soon as the relay has been energized, the contact *R*-1 closes, and then, no matter whether current continues from source *T* or not, if there is a source of current in terminal *U,* the relay will stay energized, held up. The Boolean algebra equation for the information in relay *R* is simply:

$$R = T \vee U \cdot R$$

But this is not entirely accurate, for there is a little moment of time *m* during which relay *R* is becoming energized, yet contact *R*-1 has not transferred (moved over) from the normally closed side to the normally open side of the contact. So, using the algebra of states and events *and* the delay operator, we write as the equation:

$$R = T \vee U \cdot D^m(R)$$

Let us consider a second example, Figure 13-6. At terminal *W* current is always available; it flows into relay *R* and energizes the coil, but

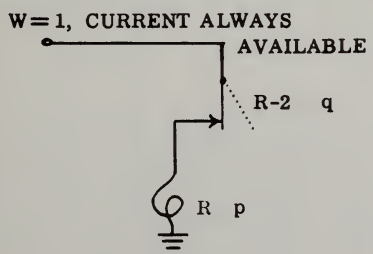


Figure 13-6. An Intermittently Operating Relay, as Used for a Buzzer.

as soon as the coil of the relay is completely energized, the core becomes a sufficiently strong electromagnet to transfer the contacts of the relay from their normally closed positions to their normally open positions. As soon as this happens, contact R-2 of relay R opens; but then current no longer flows into the coil of the relay, its magnetic field collapses, and the relay ceases to be energized. But if the relay R ceases to be energized, then contact R-2 fails to stay transferred in the normally open position, and therefore it returns to the normally closed position, whereupon the coil of the relay starts to be energized again; and so forth and so on, for an indefinite number of intermittent cycles. This kind of circuit is used electrically for buzzing an ordinary door bell.

Now let p be the truth value of "current is flowing into the coil of relay R", and let q be the truth value of "contact R-2 of relay-R is open". If we seek to write the equation for the circuit of Figure 13-6 using ordinary Boolean algebra and ignoring delay, we have

$$p = q' \text{ and } q = p, \text{ so that } p = p', \text{ which is a contradiction.}$$

However, using the algebra of states and events, we can write the equations without contradiction, as follows. Let milliseconds be our units of time and suppose that 10 milliseconds are needed for relay R to go from non-energized to energized, and vice versa. Then the equations for the circuit are:

$$p = Nq = q', \quad q = D^{10}(p).$$

These are graphed in Figure 13-7. We see that, except for isolated points where the functions are undefined, $p \vee q = 1$ and $p \cdot q = 0$, so that $p = q'$ is confirmed.

6. The HAPPEN Operator

The next one of the new operators which we may define is the operator HAPPEN or HAPPENING. Let P be a statement such as "the switch S

Figure 13-7

is on at time t", and p be the truth value of P; then we define H(P) as the statement "the switch S has been on at some time previous to t" or "it has happened that the switch S was on at some time previous to t", and H(p) as the truth value of this statement.

If p and q are two binary variables, as graphed in Figure 13-8, then H(p) and H(q) are also two binary variables as graphed in Figure 13-8.

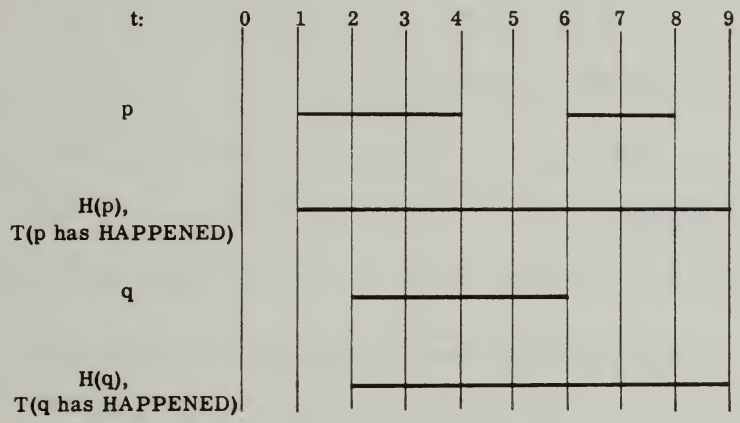


Figure 13-8. The HAPPEN Operator.

Mathematically, H(p), also written Hp, is equal to 1 for every time t after the binary variable p_t takes on the value 1; in other words, H(p_t) equals 1 if and only if there exists a time t_2 such that $t_2 < t$ and $p(t_2)$ equals 1.

There is an equation relating the DELAY operator and the HAPPEN operator. Let m be a small amount of time (a *m*oment), in particular, the smallest amount of time sufficient for an on-off circuit element of the kind being discussed to safely change from 0 to 1 or from 1 to 0 (for example, the safe operate-time of a relay). Suppose also that the duration of p is at least as long as m. Then: $H(p) = p \vee D^m H(p)$.

If I(p) equals p, the I standing for the IDENTITY operator, and if m equals 1, then we can write this equation using operators alone:

$$H = I \vee DH$$

It is useful to include with these operators the operator N for *not*. Thus we define Np as p′, equal to $1 - p$.

We can observe that we need to distinguish between "p has not happened", which may be abbreviated to NHp, and "not-p has happened", which may be abbreviated to HNp. A graphic illustration of the differences is shown in Figure 13-9.

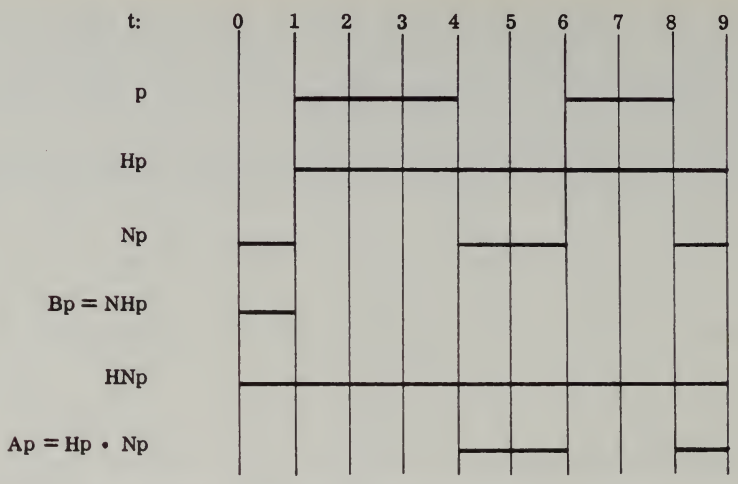


Figure 13-9. The Relation of NOT and HAPPEN, BEFORE and AFTER

7. Functional Operators in Ordinary Algebra and Calculus

In ordinary algebra, and its extensions known as differential and integral calculus, and the calculus of finite differences, functional operators like those we have just defined also appear. Here are some of them:

$$\begin{aligned}
\mathrm{E}f(x) &= f(x+1) \\
\Delta f(x) &= \mathrm{f}(\mathrm{x}+1) - f(x) \\
\mathrm{E}^k f(x) &= f(x+k) \\
\Delta^k f(x) &= \Delta[\Delta^{k-1} f(x)] \\
\mathrm{D}f(x) &= \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \\
\mathrm{D}^k f(x) &= \mathrm{D}[\mathrm{D}^{k-1} f(x)] \\
\mathrm{I}f(x) &= f(x)
\end{aligned}$$

For at least *polynomial functions* (where $f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$) many important relations of these function operators can be shown. One such relation for example is: $\Delta^k = (\mathrm{E} - \mathrm{I})^k$, and an illustration of this rule is

$$\Delta^2 f(x) = f(x+2) - 2f(x+1) + f(x).$$

Can we use these functional operators here? We can use $\mathrm{I}f(x) = f(x)$ and $\mathrm{E}f(\mathrm{x}) = f(x+1) = \mathrm{DELAY}^{-1} f(x)$. But it is not especially useful to use the other functional operators for our functions, whose only values are 1, 0, and "undefined". For example, the operator Δ applied to $f(t)$ results in 1, 0, undefined, or -1, and -1 is outside of the values of the

step-functions $f(t)$ we are dealing with. Besides, it is advantageous to us to use D for DELAY and E for ENTERING which will be explained shortly.

8. BEFORE, AFTER, ENTERING, and LEAVING

Consider the statement P:

"Circuit element E is on at time t", with truth value p.

We can consider the *state* of circuit element E, as being off at some times and on at other times; and we can consider the *events* of going from off to on, and from on to off.

Looking at Figure 13-9, we can take as an example the top line which shows the graph of p, and reports that circuit element E is on from time 1 to 4, on from time 6 to 8, and off at other times. In this case we shall have five states: three states off, two states on; and we shall have four events, at times 1, 4, 6, and 8.

With this description of conditions in front of us, we are justified in treating as equivalent the three expressions:

"before circuit element E is on"
"before binary variable p takes on its one-values"
"before p"

They are equivalent in the sense that a table of ones and zeros showing each time when the expression correctly applies, is exactly the same table for each of the expressions.

Before. Now, what is the meaning of "before p"? It means that "p has not yet happened". Let B stand for "before". Then:

$$Bp = NHp$$

After. What is the meaning of "after p"? Looking at the binary variable p in Figure 13-9, we can see that "after p" might mean "after time 4", or it might mean "after time 8", according as to whether we were thinking of after the first state of circuit element E being on, or after every state of the circuit element being on.

If p is a binary variable with only one *stretch*, only one continuous interval of time during which it has the value 1, then the state "after p" is unambiguous and it means "not p, yet p has happened". Let A stand for "after". Then:

$$Ap = Hp \cdot Np$$

For other binary variables, it is convenient to use the same definition.

Entering. The circuit element E "enters" the state of being "on" at time 1 and also enters the state of being "on" at time 6. How shall we define this?

As before, let m be the smallest amount of time sufficient for an on-off circuit element to safely change from 0 to 1 or from 1 to 0. Then a satisfactory definition of Ep, where E stands for "entering" is "p, but not yet $D^m p$":

$$Ep = p \cdot ND^m(p)$$

Leaving. Similarly, a useful definition of Lp, where L stands for "leaving" is "not-p, but still $D^m p$"

$$Lp = Np \cdot D^m p$$

Illustrations of both these definitions appear in Figure 13-10 where it has been assumed that m is about one fifth of the unit intervals in which time t is being measured.

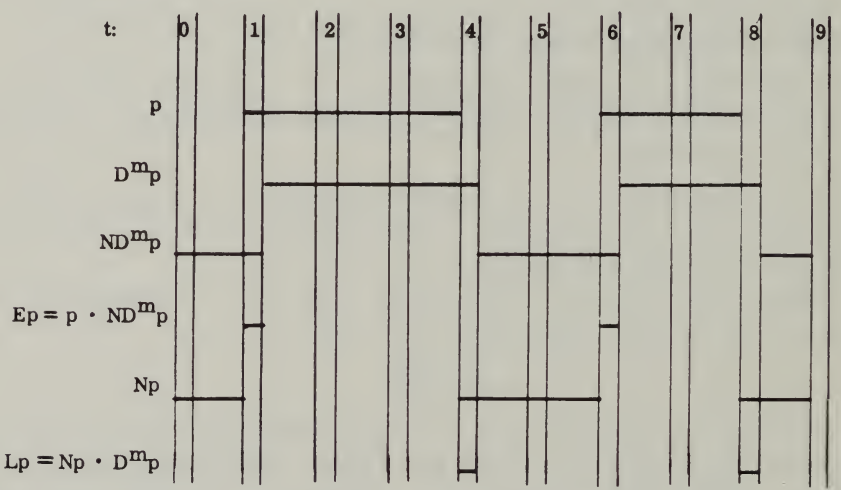


Figure 13-10. Entering and Leaving

These definitions make Ep part of p and Lp part of Np, and resolve some ambiguous cases, where p would otherwise be undefined.

We can see that as m gets smaller and smaller without reaching 0, the states Ep and Lp can still have the value 1. But in the limiting case when m equals 0, we have in both cases $p \cdot p'$, which by Boolean algebra equals 0. In dealing with circuit elements of physical hardware, the limiting case is not reached; always, at least a finite time is required

for some circuit element to influence another circuit element. Thus a customary assumption of ordinary calculus does not here apply.

9. FOR THE FIRST TIME, FOR THE SECOND TIME

If we have a binary variable which has two or more stretches, we may be interested in distinguishing between them. We shall let

I^1p mean "p for the first stretch only" or "p for the first time"
I^2p mean "p for the second stretch only" or "p for the second time"
etc.
H^2p mean "p has happened for the second time"
etc.

Can we define these in terms of DELAY, HAPPEN, and NOT? We can.

The calculation of I^1p can be made graphically as shown in Figure 13-11. There are two cases, $p_0 = 0$ and $p_0 = 1$. Let us define $Z(p)$ as 1 if

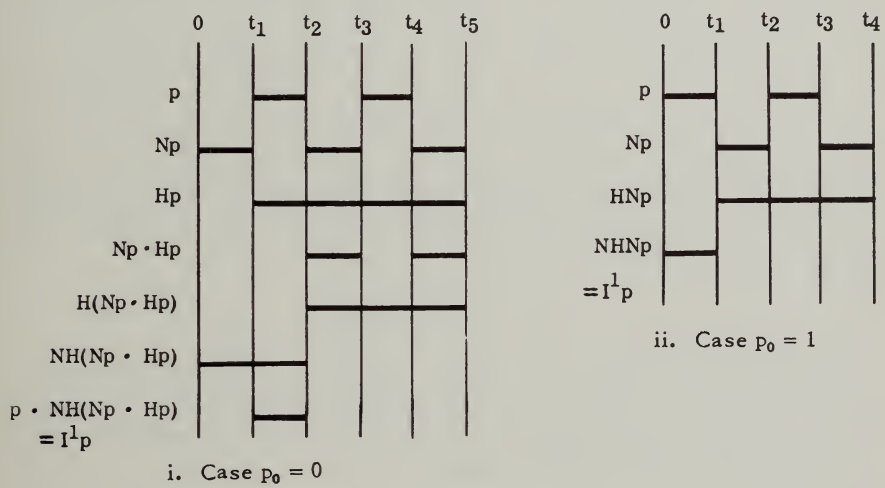


Figure 13-11. Calculation of I^1p

$p_0 = 1$ and 0 if $p_0 = 0$: we can then write both the cases in the single formula:

$$I^1(p) = [Z \cdot I \cdot NH(N \cdot H) \vee NZ \cdot NHN](p)$$

It is of course true that

$$I^1p + I^2p + I^3p + \ldots\ldots = Ip$$

for all the stretches of p taken together will equal p identically.

The calculation of H^2p in the case where $p_0 = 0$ is shown in Figure 13-12, and it is expressed in:

$$H^2p = H[I \cdot H(N \cdot H)]\ p$$

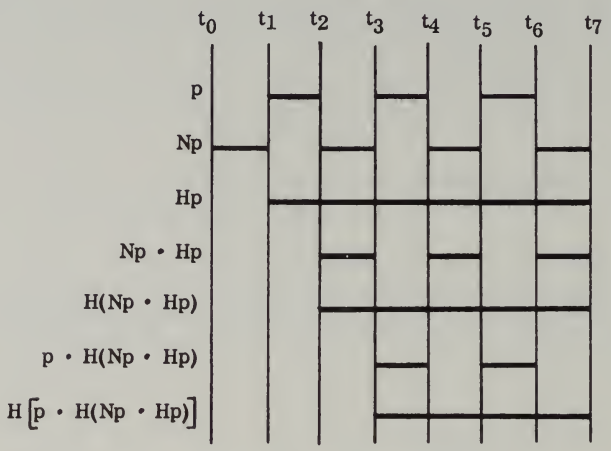


Figure 13-12. Calculation of H^2p

We could continue with a large number of the ramifications of the algebra of states and events, but probably enough has been said to show many ways in which it can be further developed.

10. Specified Times

Besides the conditions or binary variables or one-zero functions or step functions that depend entirely on states and events, there are binary functions that do not depend at all on states and events, or only partially on them. Samples of such functions are:

(a) the function which is 1 at all the time being discussed which is the same as p v Np, assuming that there are no undefined points;
(b) the function which is 1 during the alternate even-numbered tenths of a second;
(c) the function which is 1 at the time exactly 26 milliseconds after the almost instantaneous event e;
(d) the function which is 1 for exactly five seconds after the event e.

To express these functions we shall define an operator U (making use of the initial letter of universe class, an alternate name for the 1 of Boolean algebra). First, we agree that $U_a^b(p)$, where $b > a$, is equal to

the function which is 1 during the time beginning at a after the first time t_1 at which p is one and ceasing at b after t_1; this function is equal to $T(t_1 + a \leq t \leq t_1 + b)$, the truth value of "time t lies between $t_1 + a$ and $t_1 + b$." We also agree that $U_a^a(p)$ may be written at $U^a(p)$. And finally we agree that U_a^b by itself is equal to the truth value of "t is greater than or equal to a and less than or equal to b", $T(a \leq t \leq b)$.

In our discussions, the first time discussed is regularly $t = 0$ degrees and the last time discussed is ordinarily $t = 360°$, because in the first place a great number of mechanisms have a basic machine cycle, obtained from one complete rotation of a master shaft or the equivalent of this, and in the second place, since we deal with a finite and not an infinite time, a choice of time unit can be made which will place all the times we discuss in the interval from 0 to 360 degrees.

Now let us translate the four sample functions. They are respectively:

(a) U_0^{360}
(b) $U_1^2 + U_3^4 + U_5^6 + U_7^8 + U_9^{10} + \ldots\ldots$
(c) $U^{26}(e)$
(d) $U_0^5(e)$

It is understood that in the discussion in which these functions are mentioned, the context makes clear the units in which time is being measured. In the case of function (b), the *plus* (+) sign may be used in place of the *or* (v) sign, since the functions being associated do not overlap.

11. States and Events

A *state* is a condition which lasts for a time, and so is an event. But in general a *state* is a condition which lasts for a fairly long time, and an *event* is a condition which lasts for only a very short time; this is the main difference between them. But it is a relative difference, not an absolute one; for an event must not be shorter than m, the least time for a circuit element to change state safely, and it is possible for a state to be as short as that. Events, however, are likely to occur without overlapping; that is, their overlapping can ordinarily be neglected. States on the other hand are long, and they are likely to occur with overlapping between them, and their overlapping needs to be considered.

The event of CHANGING from one state p to another state r is the same as "leaving the first state, and entering the second state":

$$C(p,r) = L\,p \cdot E\,r$$

The STATE from one event e to another event f needs to be divided into two cases. In the first case, we think of the state from the first

occurrence of e to the first occurrence of f and this is "the event e has happened and the event f has not happened":

Here: $$He \cdot NHf$$

In the second case, we think of the state from any occurrence of e to the next occurrence of f. We express this as $S(e,f)$. This function has the property:

$$S(e,f) = Nf \cdot [e \vee D^m S(e,f)]$$

This is a satisfactory definition in many cases, and has many useful applications in the design of circuits. If, however, e and f are both 1 during the same period of time, S in this period is undefined. If the duration of e is shorter than m, the smallest amount of time sufficient for a circuit element to change state, then it is as if the duration of e were 0; and the same is true for f.

Chapter 14

THE ALGEBRA OF STATES AND EVENTS: APPLICATION TO SOME PROBLEMS

1. Application of the Algebra of States and Events to the Operation of Circuit Elements

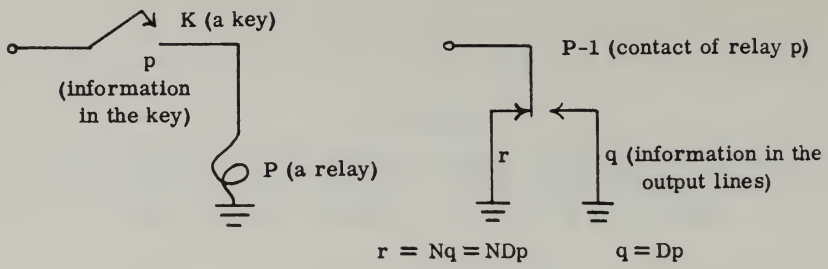
With the algebra of states and events, we can be both more accurate and more complete in our analysis of the operation of *sequential circuits*—circuits that depend for their functioning upon the sequence of states that continue for various times and of events that happen from time to time.

Although there are of course many kinds of on-off circuit elements which might be used to illustrate the application of the algebra of states and events to circuit elements, nevertheless we shall stick with relays for the same reasons as given before—their versatility, their commonness, and the fact that after one sees how to express circuits with relays, it is not very hard to translate the same kind of thinking into other kinds of circuit elements.

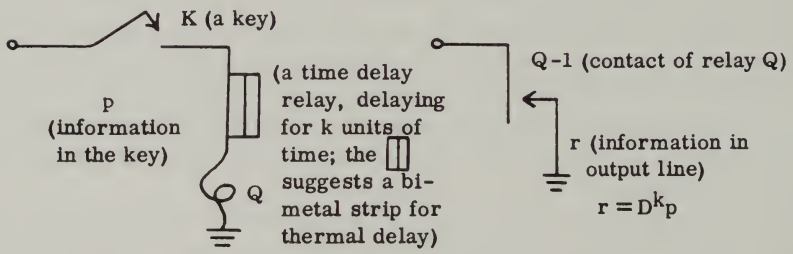
The first thing we have to do is to show how the six chief functions of the algebra of states and events, Dp, Np, $D^k p$, Hp, U_a^b, and $U_a^b(p)$ are represented in relay circuits. As before, m is the "safe-operate" time of a relay; k is any longer period of time; and the duration of any state p is not less than m.

In Figure 14-1 are shown circuits involving relays, cams, etc., which represent these functions satisfactorily. They should be studied. With combinations of these circuits, we can design many circuits.

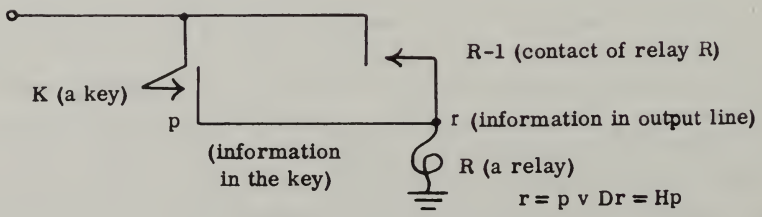
We need to talk about the mechanism illustrated in Part 4 of Figure 14-1. This is a drawing of a *cam contact*, a rotating disc with lobes. As the cam turns clockwise, the protruding lobes close the cam contact, allowing the circuit through the cam contact to be completed, providing the information q. The power for the cam could come, for example, from a continuously running shaft with a friction drive on the cam. The knob and latch restrain the cam from turning even though the shaft is turning. When the latch is moved momentarily, for example, by a solenoid electromagnet energized by the closing of the contact p, the cam is released and turns with the shaft for one revolution. In this case time from 0 to 360 degrees will be measured from the moving of the latch.



Part 1. Representation of UNIT DELAY, D, and NOT, N
(Note that unit time is required to obtain NOT in some cases)

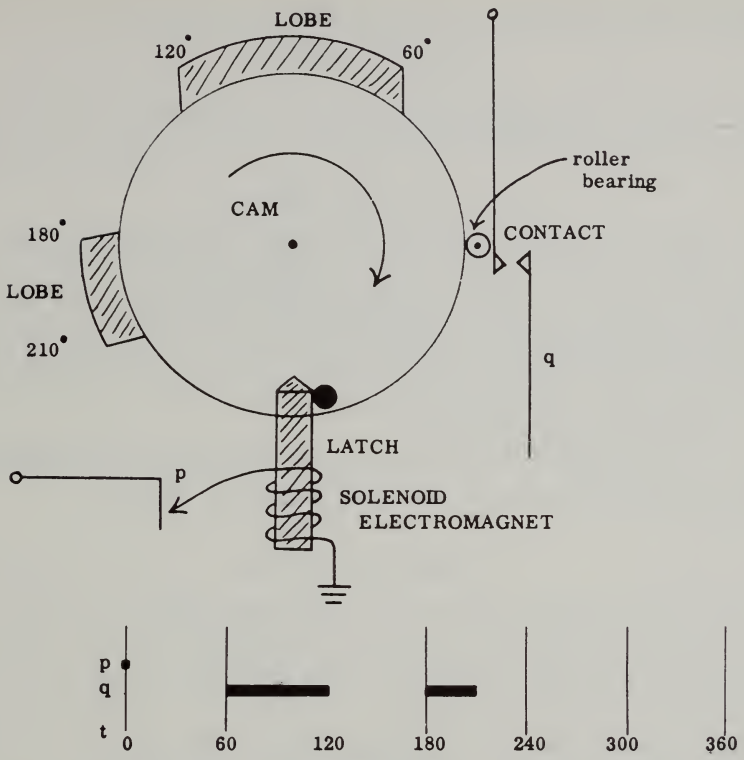


Part 2. Representation of DELAY, $D^k p$



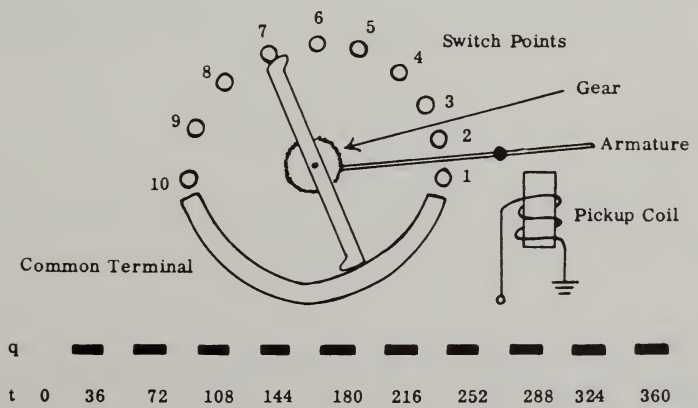
Part 3. Representation of HAPPEN, Hp

Figure 14-1



GRAPH OF p and q

Part 4. Cam Contact with Latch



Part 5. Stepping Switch

Figure 14-1 (continued)

With the latch, the cam contact expresses the function:

$$q = U_{60}^{120}(p) + U_{180}^{210}(p)$$

If we imagine the latch removed, and the cam contact continuously running, then the cam contact expresses:

$$q = U_{60}^{120} + U_{180}^{210}$$

We also need to talk about the mechanism illustrated in Part 5 of Figure 14-1. This mechanism is a *stepping switch*, a switch which "steps" from time to time depending on other conditions. As the stepping switch illustrated turns counter clockwise, the rotating arm connects terminals 1 to 10 successively with the common or transfer terminal. The release of the wiper arm to proceed one more step is permitted only when the pickup coil is energized.

We are now ready to try to apply our new algebraic instrument to solving useful problems in the design of sequential circuits.

2. A Signaling Problem

Problem: A set of circuits has a key K and three lamps P, Q, and R. Its action depends on the key K and on two conditions x and y, which are given in switch contacts. The key K is not pressed initially. The set of circuits is to have the properties:

(1) Lamp P will light if x continues after y has ceased;
(2) Lamp Q will light if K is pressed for the first time only, provided that both x and y have previously ceased;
(3) Lamp R will light if the button has been pressed twice, the condition y has never occurred, and the condition x either was on when the key K was pressed or is on when the key K is pressed

Design the circuit.

Solution: Let p equal T ("The lamp P is lighted")
Let q equal T ("Lamp Q is lighted")
Let r equal T ("Lamp R is lighted")
Let k equal T ("The key K is pressed")
(1) p IF AND ONLY IF x AND AFTER y

$$p = x \cdot Ay$$

Now $Aw = Hw \cdot Nw$, where w is any condition.
Therefore, $p = x \cdot Ny \cdot Hy$
The circuit is shown in Figure 14-2 (Part 1).

(2) q IF AND ONLY IF $I^1(k)$ AND AFTER x AND AFTER y

$$q = I^{1'}(k) \cdot Ax \cdot Ay$$
$$= I^1(k) \cdot Hx \cdot Nx \cdot Hy \cdot Ny$$

Now $I^1(w) = w \cdot NH(Nw \cdot Hw)$, where w is any condition.
Therefore q equals:

$$k \cdot NH(Nk \cdot Hk) \cdot Hx \cdot Nx \cdot Hy \cdot Ny$$

Let Z_1 (an auxiliary condition) $= H(Nk \cdot Hk)$
Then $q = k \cdot Nz_1 \cdot Nx \cdot Hx \cdot Ny \cdot Hy$
The circuit is shown in Figure 14-2 (Part 2).

(3) r IF AND ONLY IF $H^2(k)$ AND BEFORE y AND $H(E\ k$ AND $x)$

$$r = H^2(k) \cdot By \cdot H(Ek \cdot x)$$

Now $H^2w = H[w \cdot H(Nw \cdot Hw)]$, where w is any condition, and:

$$Bw = NHw$$
$$Ew = w \cdot ND^m w$$

Therefore:

$$r = H[k \cdot H(Nk \cdot Hk)] \cdot NHy \cdot H(k \cdot ND^m k \cdot x)$$

We need to discuss $k \cdot ND^m k$. We desire to pick up a relay while this combined condition exists, but this condition by definition will only last for about the time that a relay can pick up. Hence the operation of the circuit will be uncertain. So let us substitute a slower relay for $D^m k$ for which $D^{2m} k$ will apply instead of $D^m k$. Then r becomes:

$$r = H[k \cdot H(Nk \cdot Hk)] \cdot NHy \cdot H(k \cdot ND^{2m} k \cdot x)$$

We shall need for auxiliary relays:

$$Z_2 = H[k \cdot Z_1]$$
$$Z_3 = H(k \cdot ND^{2m} k \cdot x)$$

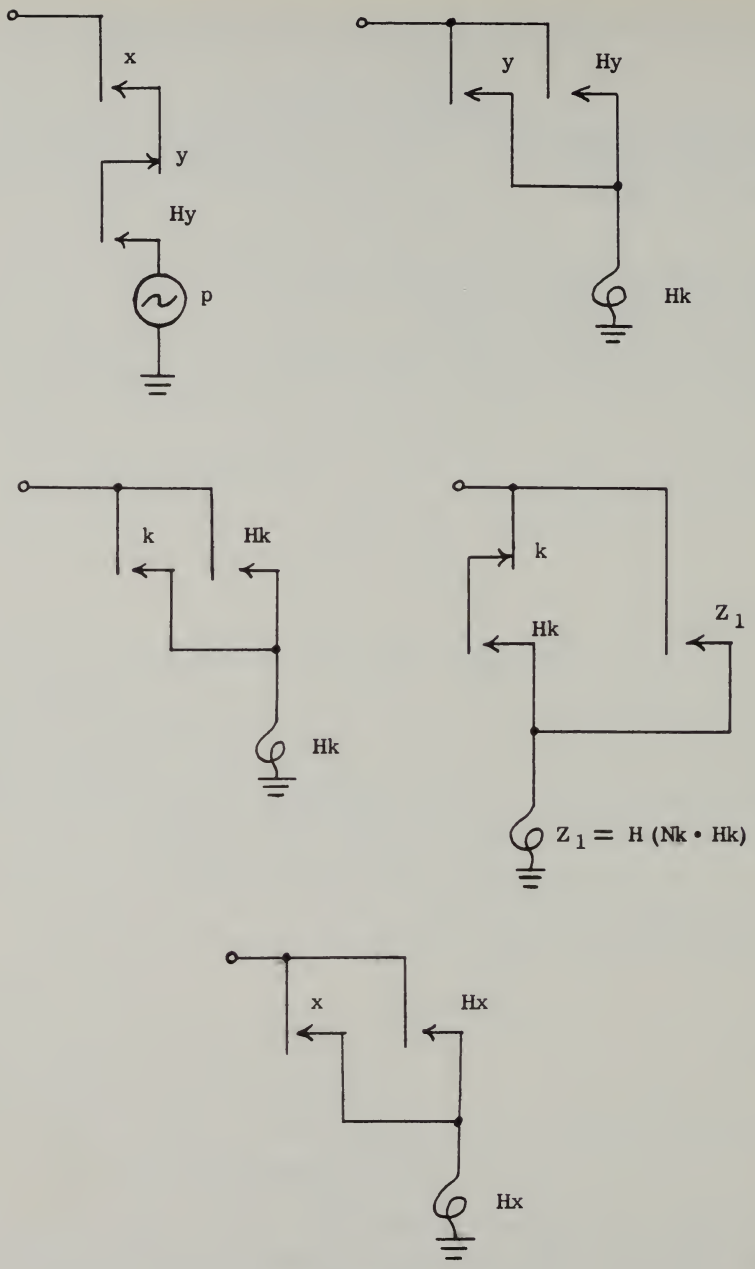
Then:

$$r = Z_2 \cdot NHy \cdot Z_3$$

The circuit is shown in Figure 14-2 (Part 3).

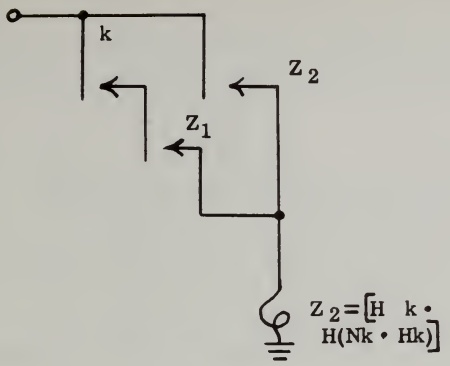
3. The Automatic Oil Furnace

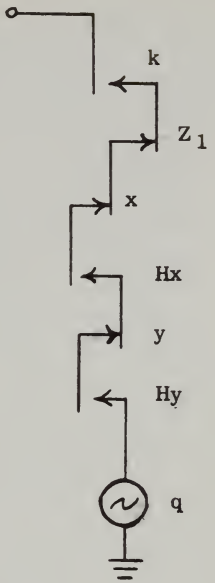
Problem: A man has an automatic oil furnace which burns oil and makes water into steam to heat the radiators in his house. The flame starts (provided no dangerous conditions exist) when the thermostat in his living room goes down to 69 degrees (j), runs at least until the end

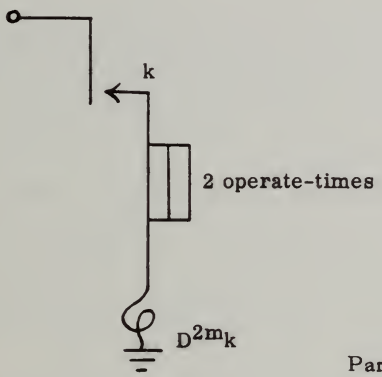


Part 1

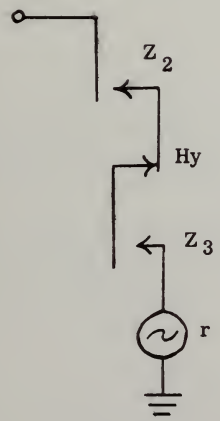
Figure 14-2

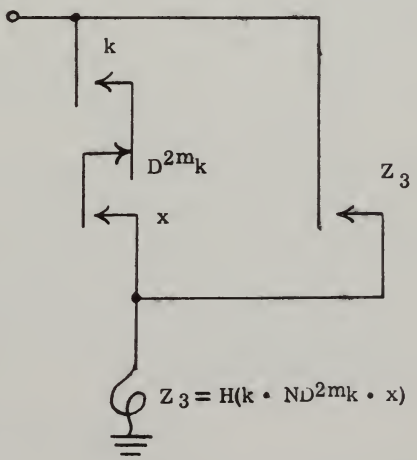






Part 2





Part 3

Figure 14-2 (continued)

of 5 minutes (m), and stops then if the thermostat has reached 72 degrees (k) or later when 72 degrees is reached. But if any of the following dangerous conditions occur, the flame in the furnace is either not allowed to light or is turned off:

the chimney is too hot (c);
the pressure in the boiler is over 8 pounds per square inch (p);
the oil in the supply tank is too low (s);
the blower that mixes air with oil and blows the mixture into the furnace is not working properly (b);
the water level in the boiler is below a certain level (w).

Design a circuit which will express the condition that the furnace flame is lighted (L).

Solution: Let the letters in parentheses in the above problem stand for the conditions there indicated. Then the desired condition is the equation for the state from one event e to whichever is later of two other events f and g (let us call this the event h, where e, f, g have special meanings which will be stated in a moment.

How shall we express h, "whichever is later of f and g"? Clearly h is equal to the event f if g has already happened and the event g if f has already happened; in symbols, "the later of f and g" $= f \cdot \mathrm{H}g \vee g \cdot \mathrm{H}f = h$. In this case f is the event 5 minutes after the thermostat's reaching a temperature of 69 degrees (m), and g is the event that the thermostat reaches a temperature of 72 degrees (k).

We can now write down the equations that will define the state we are interested in (L):

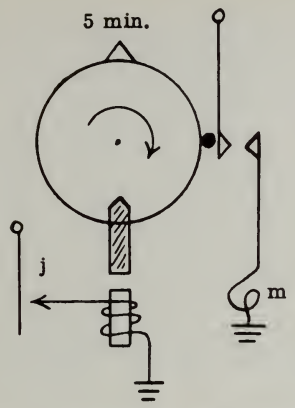
$$L = c' \cdot p' \cdot s' \cdot b' \cdot w' \cdot h' (j \vee DL)$$
$$h = m \ \mathrm{H}k \vee k \cdot \mathrm{H}m$$
$$m = U^5(j)$$

One more point: We have to modify Hk and Hm from their absolute meaning of "has ever happened"; what we need to do is to reset them (r) shortly after the time that h occurs, say 2 minutes.

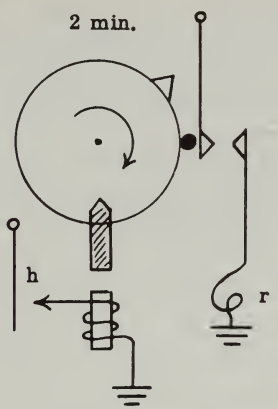
The circuits are shown in Figure 14-3. All the parts of the circuit assume that any event lasts long enough so that it can energize a relay and that the relay also will stay energized long enough so that another relay can be energized through its contacts.

4. The Responsive Traffic Light

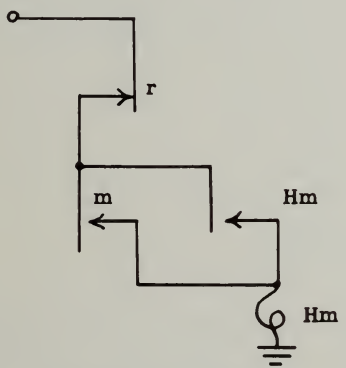
Problem: Let us take another look at the not-so-intelligent traffic light at the intersection of Lowell Ave. and Otis St., mentioned previously. It is regularly set green for Lowell Ave., and red for Otis St., because



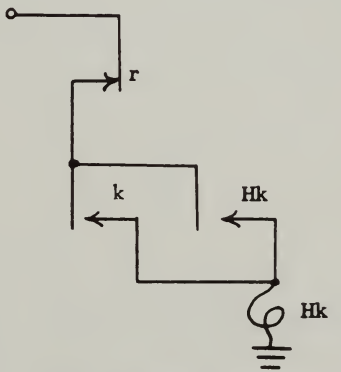
Part 1: m is the event 5 minutes from the event j
$m = U^5(j)$



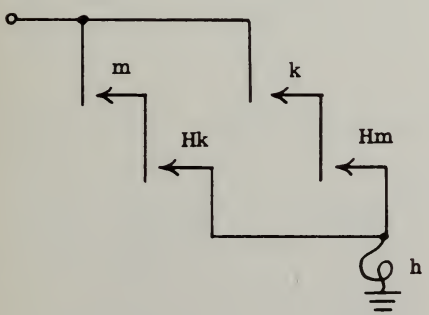
Part 2: r is the event 2 minutes from the event h
$r = U^2(h)$



Part 3: m has happened

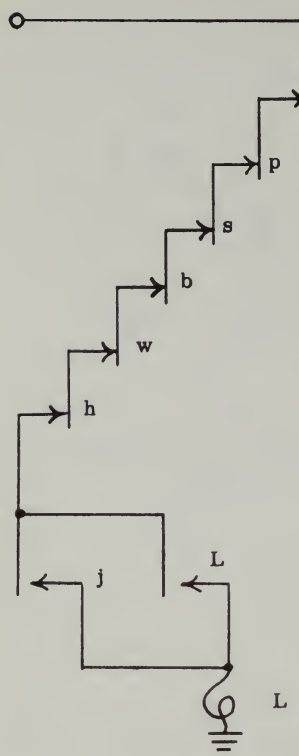


Part 4: k has happened



Part 5: the later of the events m and k, which is h
$h = m \cdot Hk \vee k \cdot Hm$

Figure 14-3



Part 6: *L*, the furnace flame is in the lighted state

Figure 14-3 (continued)

there is much more traffic on Lowell Ave. But if a car arrives on Otis St., and if at least one minute has passed with the light green for Lowell Ave., then the light changes to green for Otis St., and red for Lowell Ave., and the green for Otis St. traffic lasts for 15 seconds. (Before changing from green to red, the traffic light is yellow for two seconds.)

Design a circuit which will control the red, yellow, and green traffic lights for each street.

Solution: Here is a solution (not the only one).

We can define the states of the traffic light in the way expressed in the following table:

State	Lowell Ave.	Otis St.	Duration
L_1	green	red	indefinite
L_2	yellow	red	2 seconds
L_3	red	green	15 seconds
L_4	red	yellow	2 seconds
L_5	green	red	60 seconds

We can start counting time with the change from L_1 to L_2. During time 0 to 2 the state of the traffic light will be L_2. During time 2 to 17, the state will be L_3. During time 17 to 19, the state will be L_4. During time 19 to 79, the state will be L_5. During time 79 to ω (omega), a last time, the state will be L_1.

There will be two events. The first event is "a car arrives on Otis St., i.e., a car presses the treadle on Otis St.", which we shall designate c. The second event is "the cycle starts" (it starts with the change from L_1 to L_2) which we shall designate h.

What starts the cycle? what are the conditions that define the event h? These are (a) the arrival of a car on Otis St., if state L_1 exists; and (b) the end of 62 seconds from the start of state L_4 if during that period of 62 seconds a car has arrived on Otis St. For, clearly, if a car arrives on Otis St. during the state L_2 or L_3, it can proceed through the intersection on the green light of state L_3, and it is not necessary for the traffic light to "remember" this car and provide for it.

What will be the equations for these conditions?

The equations for the states and events will be:

$$h = c \cdot L_1 \vee U^{62}(L_4) \cdot U_0^{62} c$$
$$L_2 = U_0^2(h)$$
$$L_3 = U_2^{17}(h)$$
$$L_4 = U_{17}^{19}(h)$$
$$L_5 = U_{19}^{79}(h)$$
$$L_1 = U_{79}^{\omega}(h)$$

We can replace $U^{62}(L_4)$ by $U^{79}(h)$. (Note: we can disregard the detail that this h is the preceding h and not the current h because the circuit mechanism will treat it that way.)

The equations for the signal lights will then be:

Green, Lowell Ave. $= L_1 \vee L_5$
Yellow, " $= L_2$
Red, " $= L_3 \vee L_4$
Green, Otis St. $= L_3$
Yellow, " $= L_4$
Red, " $= L_1 \vee L_2 \vee L_5$

The circuits are shown in Figure 14-4. The first 6 cams are all mounted on the same shaft, and all unlatched by the same event h. The seventh cam is unlatched by the event c.

5. The Divorce Mill with Bigamy Alarm

Let us now consider another problem which can be dealt with in the algebra of states and events. This problem is called "The Divorce Mill

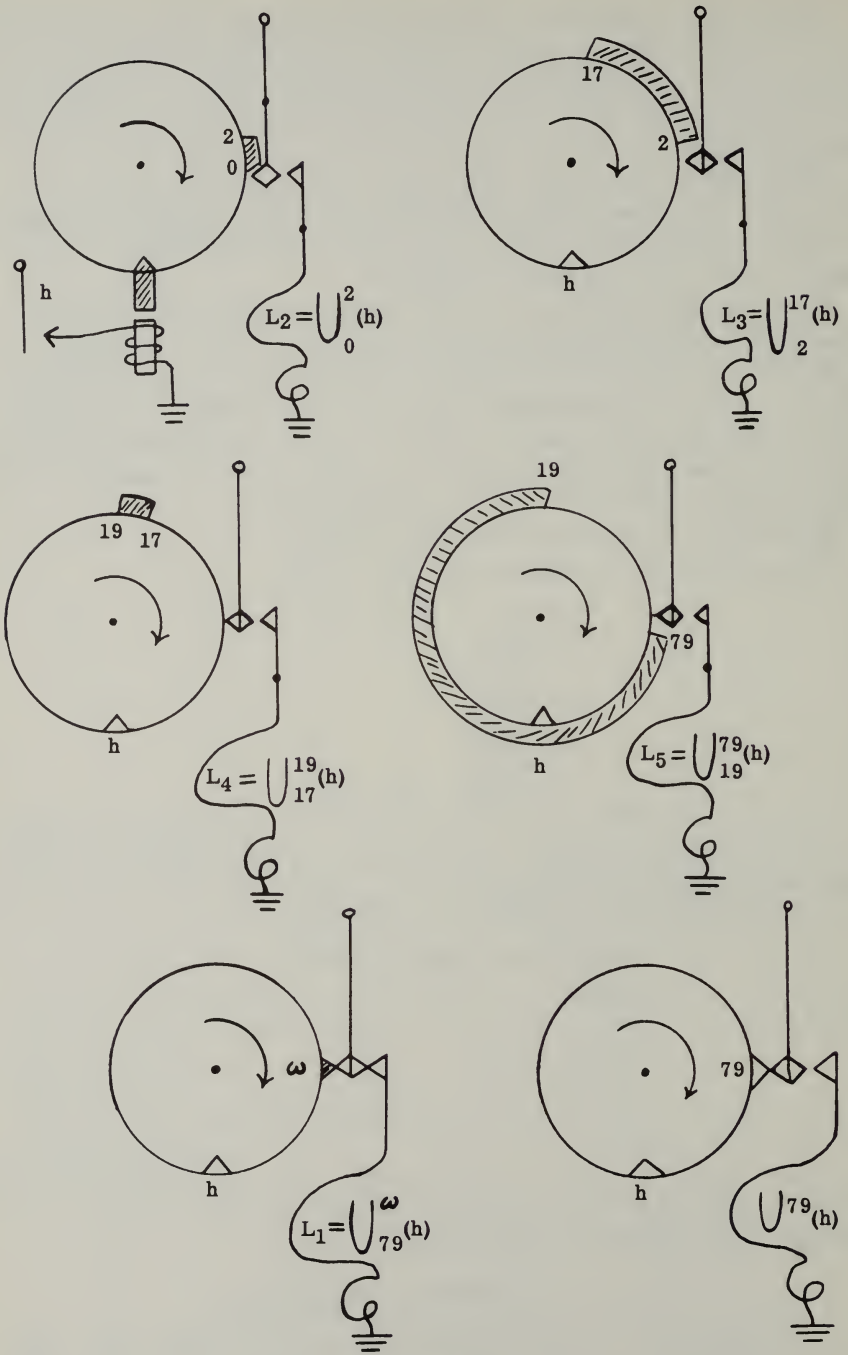

2
0
h
$L_2 = \bigcup_0^2(h)$
17
2
h
$L_3 = \bigcup_2^{17}(h)$
19 17
h
$L_4 = \bigcup_{17}^{19}(h)$
19
79
h
$L_5 = \bigcup_{19}^{79}(h)$
ω
h
$L_1 = \bigcup_{79}^{\omega}(h)$
79
h
$\bigcup_{79}^{79}(h)$

Figure 14-4

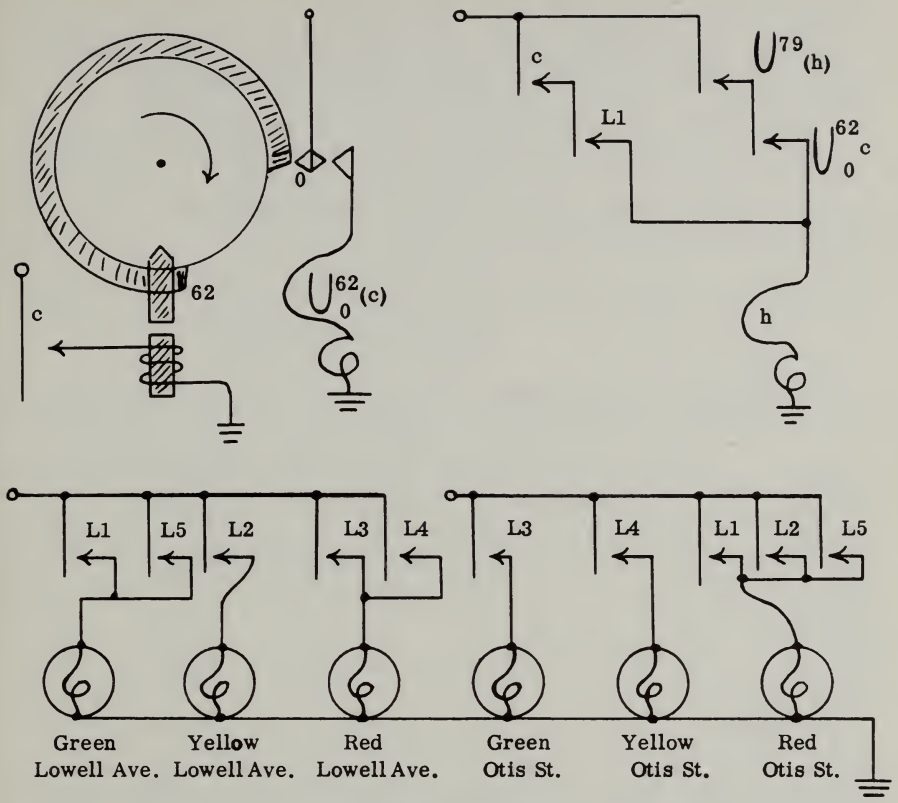


Figure 14-4 (continued)

with Bigamy Alarm." It was once actually constructed by the author and his colleagues.

Problem: Four events which can happen to a man are birth, marriage, divorce, and widowhood. Five states in which a man can therefore exist are: single; married for the first time; divorced; widowed; remarried. If a man marries a second time without attending to the petty detail of being divorced or widowed, he is in the state of bigamy.

The Divorce Mill with Bigamy Alarm has five input buttons marked "Born, Marries, Is Divorced, Wife Dies, Reset." It has five white output lamps labeled "Single, Married (for the first time), Divorced, Widowed, Remarried," and one red output lamp marked "Bigamy" and an alarm buzzer associated with it. This machine properly reports the state of a man as any of these events happen to him. Also, it properly fails to respond in any way if a wrong button is pressed; for example, if the man is in the state "divorced" and the event "born" is called for, nothing at all happens.

What should be the circuits for this machine?

Solution: The conditions which govern the circuits in this machine are stated in Tables 14-1 and 14-2. Each condition is of the form:

> "If the lamp .... is lighted, and while it is lighted, the button .... is pressed, then the resulting lamps lighted are ...."

The one letter abbreviations for the states (lamps) and the events (buttons) are shown in the table.

TABLE 14-1. CONDITIONS FOR THE DIVORCE MILL AND BIGAMY ALARM

If the lamp .... is lighted		and while it is lighted, the button ..... is pressed: Is Born (c)	Weds (n)	Divorce (e)	Wife Dies (R)	Reset (a)
		—then the resulting lamps lighted are:				
Single	s	no change	m	no change	no change	g
Married	m	"	r, b	d	w	g
Divorced	d	"	r	no change	no change	g
Widowed	w	"	r	"	"	g
Remarried	r	"	r, b	d	w	g
Bigamy	b	"	no change	no change	no change	g
Green light for starting	g	s	"	"	"	no change

From this table we can make up a second table showing for each state the events which start it and finish it.

TABLE 14-2. EVENTS STARTING AND FINISHING EACH STATE

State		Starting with	Finishing with
Single	s	$g \cdot c$	$n \vee a$
Married	m	$s \cdot n$	$n \vee e \vee k \vee a$
Divorced	d	$(m \vee r) \cdot e$	$n \vee a$
Widowed	w	$(m \vee r) \cdot k$	$n \vee a$
Remarried	r	$(m \vee d \vee w) \cdot n$	$e \vee k \vee a$
Bigamy	b	$(m \vee r) \cdot n$	a
Green light for starting	g	$(s \vee m \vee d \vee w \vee r \vee b)\, a$	c

In an earlier section we showed that the general equation for the state S from one event e to another event f was:

$$S = f' (e \vee D\, S)$$

Using this general equation we can write down the equation for the state *single*, s:

$$s = (n \vee a)' (gc \vee Ds)$$
$$= n'a' (gc \vee Ds)$$

Similarly,

$$m = n' \cdot e' \cdot k' \cdot a' (sn \vee Dm)$$
$$d = n'a' [(m \vee r)\, e \vee Dd]$$
$$w = n'a' [(m \vee r)\, k \vee Dw]$$
$$r = e'k'a' [(m \vee d \vee w)\, n \vee Dr]$$
$$b = a' [(m \vee r)\, n \vee Db]$$
$$g = c' [(s \vee m \vee d \vee w \vee r \vee b)\, a \vee Dg]$$

Now in order to make the circuits corresponding to these equations work properly, let us think about the operate-times and drop-out times of the six relays corresponding to the states s, m, d, w, r, g.

For example, suppose that the relay g operates and drops out the fastest, in 5 milliseconds, and relay s operates and drops out the slowest, in 8 milliseconds. Let us consider the time-graphs for the variables g, c, s, Ds in the equation for s: $s = n'a' (gc \vee Ds)$. Suppose that any pressing of a button, any of the events c, m, e, k, a, lasts at least 10 milliseconds, safely longer than the operate-time or drop-out time of any relay.

Now, what will happen if the event c ("is born") starts at time 1 (measured in milliseconds), and lasts until time 11? The event c happening at time 1, while the state g exists, causes the relay g (green light) to start to drop out, because this event is the end of the state g. Since g operates or drops out in 5 milliseconds, by time 6, relay g is completely dropped out. At some time between time 1 and time 6, say half way between, at time 3½, current can no longer flow through a contact of relay g. Consequently, at time 3½ it is no longer possible to obtain the function $g \cdot c$. The relay s therefore starts to pick up for 2½ milliseconds, from time 1 to time 3½, but then voltage ceases, and it proceeds to drop out. The hold contact Ds of the relay s of course would not start to hold the relay up until time 9, since the operate-time of relay s is 8 milliseconds. So the equation $s = n'a' (gc \vee Ds)$ will not work. (The situation we have just described is graphed in Part 1 of Figure 14-5.)

What then must we do?

The remedy is to associate with each of the six relays

$$u = s, m, d, w, r, q$$

a second relay which will express D^9u, where the D^9 refers to delay for

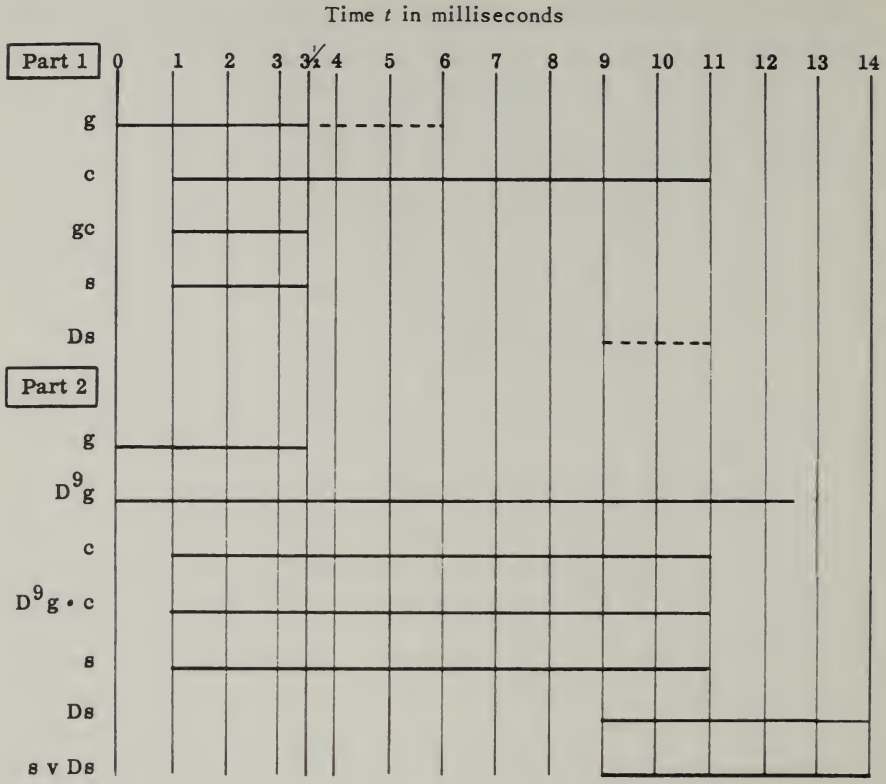


Figure 14-5

9 milliseconds, and to make use of the fact that D^9u will not drop out until 9 milliseconds following the beginning of the dropping out of relay u. Then, for each of the six states u in the six equations above, we substitute D^9u. Then our circuits should operate satisfactorily.

To check this statement, let us take a look at Part 2 of Figure 14-5. As before, suppose that event c ("is born") starts at time 1 millisecond. Relay g will start to drop out, and its contacts we assume will open at time 3½. But now, relay D^9g is still energized, and will remain energized for 9 milliseconds after the g contacts open. Consequently D^9g will be in the state *on* until time 12½. The function $D^9g \cdot c$ can be obtained until the end of c at time 11. The function $s \vee Ds$ therefore can be obtained from millisecond 1 to the end of the times we are considering.

There is another contradiction we must remove from the general equations: in the equation for m, the logical product of n and n' appears, and this of course is zero. The trouble lies in the difference between pressing the button "weds" (the event n) the first time, and pressing the

button not for the first time but for the second or a later time. Let:

n_1 = the event pressing the button for the first time
n_2 = the event pressing the button for a second or later time

Using $I^1 p = p \cdot \mathrm{NH}\ (Hp \cdot Np)$ we have:

$$n_1 = n \cdot \mathrm{NH}\ (Hn \cdot Nn)$$
$$n_2 = n \cdot \mathrm{H}\ (Hn \cdot Nn)$$

The circuit equations with this correction become:

$$s = n_1' \cdot a'\ (D^9 g \cdot c \vee Ds)$$
$$m = n_2' \cdot e' k' a'\ (D^9 s \cdot n_1 \vee Dm)$$
$$d = n_2' \cdot a'\ [(D^9 m \vee D^9 r)\ e \vee Dd]$$
$$w = n_2' \cdot a'\ [(D^9 m \vee D^9 r)\ k \vee Dw]$$
$$r = e' k' a'\ [(D^9 m \vee D^9 d \vee D^9 w)\ n_2 \vee Dr]$$
$$b = a'\ [D^9 m \vee D^9 r)(n_2 \vee Db]$$
$$g = c'\ [D^9 s \vee D^9 m \vee D^9 d \vee D^9 w \vee D^9 r \vee D^9 b)\ a \vee Dg]$$

The circuits are shown in Figure 14-6.

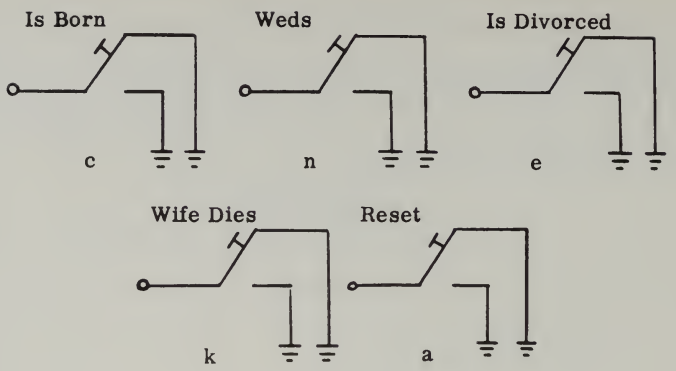
6. A Robot Animal

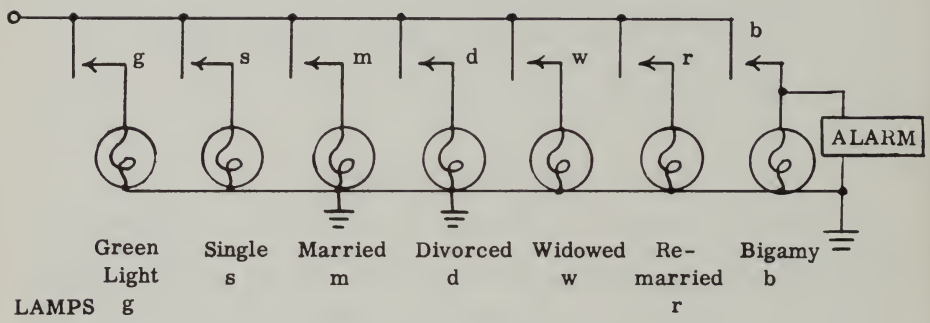
The problem we have just finished is characteristic of programming problems—problems where we desire to program a piece of apparatus, a robot for example, to behave in specified ways. To operate such a piece of apparatus often means to arrange (1) a number of states for it, and (2) changes from one state to another state when certain events happen. Then each state determines a certain behavior or activity.

A robot is a machine which is able to "behave" by itself; thus it is more than just an automaton, a machine which moves by itself. A robot is able to take in sensations by means of sensing devices, perform actions by means of acting devices, and correlate sensations and actions by means of circuits or mechanisms which express one or more sets of instructions. This last class of hardware within a robot is a rudimentary or advanced apparatus for "thinking."

As stated above, the activity of a robot may be divided into a number of types of behavior, which it engages in one at a time. For example, Squee, an electronic robot squirrel constructed by the author and his colleagues in 1950-51, and described in *Radio Electronics*, December 1951, has three activities:

(1) "Hunting"—searching around its neighborhood for a "nut" (a tennis ball), which is "picked up" in its "hands" (two halves of a scoop at the front of the robot, which can open and close);


Is Born
Weds
Is Divorced
c
n
e
Wife Dies
Reset
k
a


g
s
m
d
w
r
b
ALARM
Green Light
Single
Married
Divorced
Widowed
Re-married
Bigamy
LAMPS g
s
m
d
w
r
b

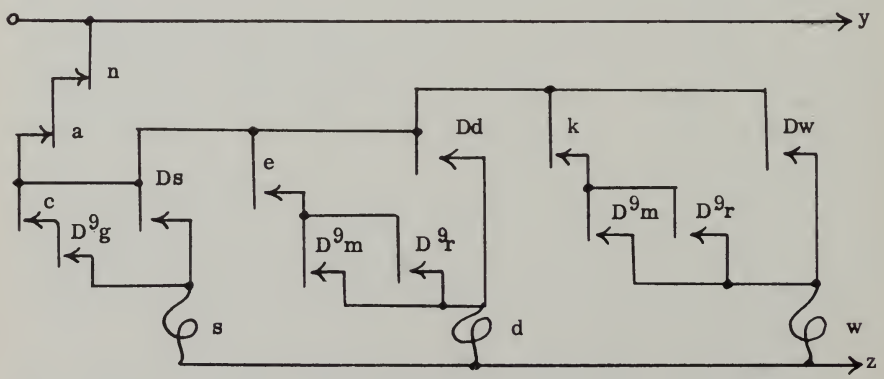

y
n
a
Ds
c
D9g
e
D9m
D9r
Dd
k
D9m
D9r
Dw
s
d
w
z

Figure 14-6

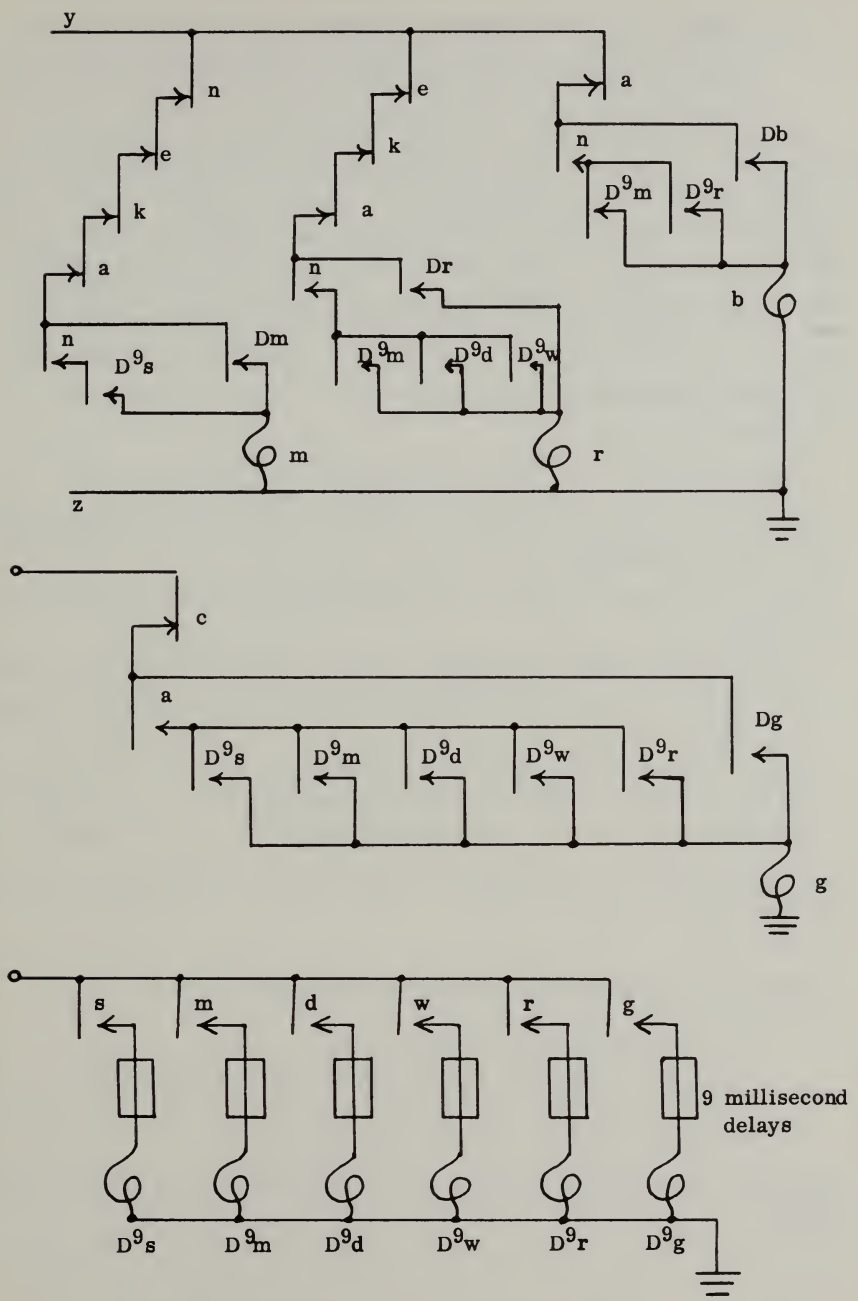


Figure 14-6 (continued)

(2) "Homing"—proceeding to its "nest" (a piece of flat metal);
(3) "Depositing"—opening its "hands" upon arrival at the nest, thus allowing the nut to roll out, and, at the same time, backing away, thus leaving the nut at the nest.

These three activities followed each other; the first two were of indefinite length of time, depending on events; the third activity was exactly five seconds long. Each activity is governed by what may be called an *activity relay* or *program relay*: (1) the pattern of sensations and conditions which starts the program energizes the relay; (2) while that activity continues, reading through the contacts of that relay permits the various items of behavior for that program to take place; (3) the pattern of sensations and conditions which stops the program causes that relay to drop out.

Let us consider the problem of a simple robot animal.

Problem: Design a robot which simulates searching for food, eating, sleeping, and escaping from danger. What would be the circuits which would control the robot?

Solution: It would be logical to decide upon some rules including the following:

(1) No matter when the signal "danger" arrives, immediately the robot goes into the danger-avoiding program until a signal "safe" arrives.
(2) Otherwise, the robot normally is seeking food, and when it finds food, it eats until it is "no longer hungry."
(3) When it is "tired", it will go to sleep, until it "wakes up."

The conditions for controlling this robot may be expressed in Tables 14-3 and 14-4.

The equations for the robot animal will then be:

$$d = q' \cdot a' \, [D^2 h \vee D^2 e \vee D^2 s) \, p \vee Dd]$$
$$h = p' \cdot f' \cdot t' \, [D^2 d \cdot q \vee D^2 s \cdot w \vee (D^2 d \vee D^2 e \vee D^2 s) \, a \vee Dh]$$
$$e = p' \cdot t' \cdot i' \cdot a' \, [D^2 h \cdot f \vee De]$$
$$s = p' \cdot w' \cdot a' \, [(D^2 h \vee D^2 e) \, t \vee D^2 e \, i \vee Ds]$$

They can be put into circuit diagrams in the same way as the preceding problem.

TABLE 14-3. CONDITIONS FOR THE ROBOT ANIMAL

If the state of the animal is	and if the event .... occurs: Danger (peril) *p*	Finding Food *f*	Is Sleepy (tired) *t*	Safe *q*	Is no Longer Hungry *i*	Wakes Up *w*	Reset *a*
	then the state of the animal changes to						
Avoiding danger *d*	no change	no change	no change	*h*	no change	no change	*h*
Hunting food *h*	*d*	*e*	*s*	no change	no change	no change	no change
Eating *e*	*d*	no change	*s*	no change	*s*	no change	*h*
Sleeping *s*	*d*	no change	no change	no change	no change	*h*	*h*

TABLE 14-4. EVENTS STARTING AND FINISHING EACH STATE

State		Starting with	Finishing with
Avoiding danger	*d*	$(h \vee e \vee s)\, p$	$q \vee a$
Hunting food	*h*	$dq \vee sw \vee (d \vee e \vee s)\, a$	$p \vee f \vee t$
Eating	*e*	hf	$p \vee t \vee i \vee a$
Sleeping	*s*	$(h \vee e)\, t \vee ei$	$p \vee w \vee a$

Chapter 15

SYMBOLIC LOGIC AND THE PROGRAMMING OF AUTOMATIC COMPUTERS

1. Logical Operations in Programming

As long as we do mathematics with pencil and paper, and perhaps desk calculating machines, we notice the numerical operations in a calculation; but we tend to carry out the logical operations in our heads or on paper and do not notice them. As soon as we begin to use automatic computers that perform calculations in long sequences, however, we find out quickly that we must pay attention to nonnumerical reasoning operations, as well as to numerical operations; and we find it helpful to express logical operations in symbols. With these symbols, we grasp a power that we did not have before. Furthermore, these symbols can often be made to fit into ordinary mathematical language. Then a computer routine is no longer half flesh and half fowl; the whole routine takes on the same mathematical, logical, symbolic nature.

2. The Basic Logical Operations that Automatic Computers Should Perform

At least the following logical operations need to be provided for in automatic computers.

(1) *Selection*

A computer should, depending on an indication p, be able to select one number a or another number b. The algebraic equation for the operation *selection* is

$$c = ap + b(1 - p)$$

Here a and b are numbers and p is a binary variable that can be either 1 or 0. If p equals 1, c equals a; if p equals 0, c equals b.

(2) *Comparison*

A computer should be able to compare two numbers and decide whether they are equal or not equal. The algebraic equation for the operation of comparison is:

$$p = \mathrm{T}\ (a = b)$$

or equivalently $q = \mathrm{T}(a \neq b)$. If a equals b, then p equals 1 and q equals 0. If a is not equal to b, then p equals 0, and q equals 1.

(3) *Checking Against a Tolerance*

A computer should be able to determine whether a certain difference d is in absolute value less than a certain positive tolerance t. The algebraic equation for this operation is:

$$p = \mathrm{T}(|a| < t)$$

In words, p is equal to 1, if the absolute value of d is less than t; if not, p is equal to 0. This is the criterion often used for an automatic computer to determine whether a loop of instructions should be repeated once more, or dropped.

(4) *Sequencing*

A computer should be able to put numbers in sequence. In other words, a computer should be able to compare two or more numbers presented on two or more input tapes, determine which is the smallest (or largest), and copy that number on an output tape. In this way, two or more tapes of numbers each in sequence may be collated into a single tape in sequence. The algebraic equation for this operation (which needs to include the instruction what to do if the two numbers presented are equal) is:

$d = a \cdot \mathrm{T}(a < b) + b \cdot \mathrm{T}(b < a) + c \cdot \mathrm{T}(a = b)$, where c is usually either a or b.

Other logical operations that may be found in some automatic computers, or that may need to be built or programmed into truly versatile automatic computers, are matching, merging, tabulating, sorting, implication, denial, exception, etc.

3. Symbolic Logic to Express the Operation of a Counter Mechanism

An interesting example of the fusion of mathematics and logic in computers is found in a punched card machine known as a tabulator made by International Business Machines Corp. A tabulator is a machine that either lists the information contained in a series of punched cards, or prints totals derived from them, or does both.

Among other devices a tabulator contains a counter mechanism that can handle numbers of six decimal digits. Its operation is controlled by a plugboard, an assembly of plugwires (or patch cords) and connecting hubs (or terminals), all placed in a frame. This wired-up frame ex-

presses the instructions or program for a tabulator problem, and is changed from problem to problem.

If we examine the plugboard frame of a tabulator, we find 4 inputs and 1 output for the counter mechanism. The 4 inputs are:

(1) A set of 6 hubs, one for each digit, called "Counter Entry", which takes in the 6 digit number a.
(2) A single hub, called the "Add" hub, which takes in a pulse p.
(3) A single hub, called the "Subtract" hub, which takes in a pulse q.
(4) A single hub, called the "Counter Total Control" hub, which takes in pulse r.

Logically a pulse can be only 1, the presence of a pulse, or 0, the absence of a pulse; but actually in the machine the timing of the pulse is also important. The output is a set of 6 hubs called "Counter Total Exit", which puts out a number b.

We can express the operation of the counter mechanism with two simple algebraic equations. Let the number held in the counter at any time be h. Then at any cycle x,

$$h_x = h_{x-1}(1 - r_{x-1}) + a_x(p_x - q_x) + 999{,}999\, p_x q_x$$

and

$$b_x = h_x r_x$$

What do these two equations mean? If just p is impulsed, the counter adds a. If just q is impulsed, the counter subtracts a. If both p and q are impulsed, the counter adds 999,999. If r is impulsed, the counter total exit reads out the number held in the counter, and the counter is at the same time cleared.

Obviously, the mechanism would be more flexible if we could read out the number in the counter without necessarily clearing. In fact, IBM provides a modified counter mechanism with which this is possible.

Since a, b, and h are variables that can be regular numbers, this department belongs to mathematics. But p, q, and r are binary variables that can have only the values 1 or 0, like "yes" or "no", and this department belongs to symbolic logic. Only when we fuse the two departments can we exactly express the operation of the counter mechanism.

4. Other Relations of Symbolic Logic to Automatic Computers

The relation of symbolic logic to automatic computers is, however, far more extensive than just the combining of symbols of symbolic logic and symbols of mathematics in the same equations. In programming automatic

computers, we notice a number of important questions, all belonging in the territory of symbolic logic rather than in mathematics.

Here are some of the questions:

(1) What is a sufficient set of commands to instruct an automatic computer to solve any possible problem in mathematics or reasoning, for which a feasible method of solution is known?
(2) How do you instruct a computer with a program so that the computer itself can develop and use other programs?
(3) What is the most efficient way to give an automatic computer instructions or commands?

Some of these questions are obviously difficult, and may not be answered for many years. Some of them can be answered to some extent now.

Let us take an actual problem and see how questions of logic actually occur in it.

5. Finding Square Root

A problem that will require several subroutines is the one of finding the square root of a number using an *iterative formula*—one that gives a better result each successive time that we apply it. One such formula for square root is

$$x_{n+1} = 1/2(x_n + Y/x_n),$$

where Y is the number for which we want the square root, and the x's are successive approximations. Each time we apply this formula we get a better approximation to the true square root. We begin by making any kind of rough guess about the square root of the number Y, and we call this first rough guess x_1.

To test the procedure, let us obtain the square root of 67.2. We choose 8 as a first guess, because 8 times 8 is 64, and 9 times 9 is 81, and 67.2 is in between these results. So 8 is our first approximation, x_1. In Round 1, 8 divided into 67.2 gives 8.4. The average of 8 and 8.4 is 8.20. This is x_2, our second approximation. It is correct to 3 figures. In Round 2, 8.20 divided into 67.2 gives 8.195122. The average of this number and 8.20 is 8.197561. This is x_3, our third approximation. It is correct to 6 figures. The result of the next round is 8.1975606125, x_4. This is correct to 10 figures. So we see that, with a reasonable guess and two or three divisions, we can obtain all the accuracy we can ordinarily use.

Good iterative formulas are like this: they approximate the true value quickly, and they are very useful on automatic computers. To perform

this problem on a machine we recognize 5 subroutines: the subroutine for reading data from input into storage; for carrying out the formula once; for deciding whether to repeat the formula for another round; for preparing to repeat; and the subroutine for sending the answer from storage to output, and stopping. The third subroutine, deciding whether to repeat, is purely logical.

6. Automatic Computer ZAC

Let us imagine and stipulate a simple automatic computer (ZAC, the "Z Automatic Computer"), able to do this problem.

ZAC has an input tape, an output tape, and 70 registers for storage ordinarily of 8 decimal digits. ZAC has a calculating unit, or "computer", and this can take in an operation *OP* on one channel, take in 2 numbers a and b on 2 more channels, and give out the result c on the fourth channel:

$$c = a\ OP\ b.$$

ZAC has a program register, which holds each successive instruction that governs the machine. We can transfer numbers or orders into and out of the program register. Some, but not all, numbers will have meaning as orders to ZAC. The program register regularly holds 5 digits: the first 2 digits will be the number of the order, n; the middle digit will be the kind of the order, k; and the last 2 digits will ordinarily be the number of a register, r. At any cycle, the order, or instruction, n, k, r, stored in the program register tells the machine what it is to do at that cycle.

The kind-of-order numbers, k, we shall suppose, can vary from 0 to 9. Using these 10 numbers, we have sufficient flexibility to tell the machine all that we want it to do in finding square root.

The register numbers r vary from 10 to 79. Registers 10–49 will usually store orders having the same order number as the register number; registers 50–69 will usually store numbers in the calculation; and registers 70–79 will usually store constants.

The most interesting of these sets of numbers is the kind-of-orders, k. Every k is followed by an r. This is the meaning:

If k is 0, the machine transfers the operation stored in register r to the computer (the calculating unit) and goes to the next order numerically.

If k is 1, the machine transfers the number in register r to computer register a and goes to the next order numerically.

If k is 2, the machine transfers the number in register r to computer register b and goes to the next order numerically.

If k is 3, the machine transfers the computer result c to register r and goes to the next order numerically.

If k is 4, the machine transfers the number on the input tape to register r and goes to the next order numerically.

If k is 5, the machine transfers the number in register r to the output tape and goes to the next order numerically.

If k is 6, the machine is instructed to go to order r (obtaining it from register r), instead of (as the machine normally does) going to the next order numerically.

If k is 7, and if the number in register 69 is 1, the machine is instructed to go to order r; if k is 7, and if the number is not 1, the machine goes to the next order numerically.

If k is 8, and if the number in register 69 is 1, the machine is instructed to go to the order number stored in register r; if k is 8, and if the number is not 1, this order tells the machine to go to the next order numerically.

If k is 9, the machine stops.

Some of the register numbers that may follow the kind-of-order $k = 0$ are 70, 71, 72, 73, 74, 75. These registers contain signals that set the computer for 6 operations, respectively:

70, transfer,	$c = a$
71, addition,	$c = a + b$
72, subtraction,	$c = a - b$
73, multiplication,	$c = a \cdot b$
74, division,	$c = a \div b$
75, inequality,	$c = T(a \neq b)$

7. The Program for Square Root

The program for square root using ZAC is shown in Table 15-1.

The first subroutine consists of orders 10, 11, 12. Here we read out from the input tape into registers of the machine. Then we proceed to order 19.

Subroutine No. 2 is now carried out. In orders 19–29, we cover the division, the addition, and the multiplication required by the iterative formula. Then we go to order 32.

In subroutine No. 3, in orders 32–35, we cover inequality; we test 2 successive approximations to see if they are equal or unequal. If they are unequal, we record a 1 in register 69. (In practice, a difference less than a certain tolerance would be accepted as equality.)

Now we come to a choice of program. Using order 36, we go to order 38 if, and only if, there is a 1 in register 69; in other words, if x_n and x_{n+1} are unequal. If there is a 0 in register 69—in other words, if the last two x's are equal—then we go to the next order, 37, and that routes us to order 43.

TABLE 15-1. FIVE-DIGIT ORDER (STORED IN REGISTERS 10–44)

Subroutine	Number, n	Kind, k	Register, r	Meaning
1. Reading data from input into storage	10	4	50	Input of Y to 50
	11	4	56	Input of x_1 to 56
	12	4	51	Input of $\frac{1}{2}$ to 51
	13	6	19	Go to order 19
2. Carrying out the formula once	19	0	74	*Division* to computer
	20	1	50	Y to computer
	21	2	56	x_n to computer
	22	3	52	Y/x_n to 52
	23	0	71	*Addition* to computer
	24	1	52	Y/x_n to computer
	25	3	53	$x_n + Y/x_n$ to 53
	26	0	73	*Multiplication* to computer
	27	1	53	$x_n + Y/x_n$ to computer
	28	2	51	$\frac{1}{2}$ to computer
	29	3	57	$\frac{1}{2}(x_n + Y/x_n) = x_{n+1}$ to 57
	30	6	32	Go to order 32
3. Deciding whether to repeat or not	32	0	75	*Inequality* to computer
	33	1	56	x_n to computer
	34	2	57	x_{n+1} to computer
	35	3	69	$T(x_n \neq x_{n+1})$ to 69
	36	7	38	Go to order 38 if 1 is in 69
	37	6	43	Go to order 43
4. Preparing to repeat	38	0	70	*Transfer* to computer
	39	1	57	Old x_{n+1} to computer, equal new x_n
	40	3	56	New x_n to 56
	41	6	19	Go to order 19
5. Sending answer from storage to output and stopping	43	5	57	x_{n+1} in 57 to output
	44	9	44	Stop

In orders 38–40, we remove x_n from the iterative formula, subroutine No. 2, and insert x_{n+1} instead. Then with order 41 we go to subroutine 2, which will now compute the next approximation.

In order 43 we read out the final value of x into the output tape, and with the next order stop the machine.

Thus we see how we can program square root with a machine.

In this program, we have to recognize the operation of inequality: this is logic rather than mathematics. We have to recognize different subroutines: this is logic rather than mathematics. We have to provide for the branching of instructions: this is logic rather than mathematics. We have to provide for the machine's deciding for itself when it will stop using a formula and, instead, give out the answer: this, too, is logic rather than mathematics.

We can anticipate that there will be more and more fusion between numerical mathematics, on the one hand, and nonnumerical reasoning, or symbolic logic, on the other. Machines that play games, machines that separate true combinations of statements from false combinations, other kinds of information-handling machines where emphasis is on logical competence rather than on mathematical competence, are already in existence. Symbolic logic, intelligent machines, and mathematics will continue to enrich one another in many significant ways.

SELECTED BIBLIOGRAPHY

1. SYMBOLIC LOGIC: HISTORICAL

BOOLE, GEORGE, *Investigation of the Laws of Thought*, London, Eng: Walton, 1854; reprinted many times, including a reprint by Dover Publications, New York.
Boole's great contribution.

BOOLE, GEORGE, *The Mathematical Analysis of Logic*, Cambridge, England, 1847; reprinted by Oxford Press, New York, 1948.

VENN, JOHN, *Symbolic Logic*, London, Eng: Macmillan & Co., 1894.
Includes 19 problems and solutions, on pages 331–59.

DODGSON, C. L. (Lewis Carroll), *Symbolic Logic, Part I, Elementary*, 4th edition, London, Eng: Macmillan & Co., 1897, 232 pp; reprinted 1955 by Berkeley Enterprises, Newtonville, Mass.
Contains Lewis Carroll's inimitable and entertaining problems in symbolic logic, his method of solution (now partly out of date), and his sketches of Parts II and III, which he never wrote, since he died in 1898.

HUNTINGTON, E. V., "Sets of Independent Postulates for the Algebra of Logic," in *Transactions of the American Mathematical Society*, V, (1904), pp 288–309.
The source of the postulates given in Chapter 6.

RUSSELL, BERTRAND, and A. N. WHITEHEAD, *Principia Mathematica*, 3 vols., Cambridge, Eng: The University Press, 1910–13.
A great landmark in the attempt to place the foundations of mathematics upon a sound logical basis of a few given undefined concepts and a few given unproved assertions.

SHANNON, CLAUDE E., "A Symbolic Analysis of Relay and Switching Circuits," in *Transactions of the American Institute of Electrical Engineers*, vol. 57, 1938, pp 713–723; reprinted, 1952, by Berkeley Enterprises, Newtonville, Mass.
The first application of Boolean algebra to relays and on-off circuit elements.

2. SYMBOLIC LOGIC: THE MODERN PRESENTATION

PFEIFFER, JOHN E., "Symbolic Logic," in *Scientific American*, December, 1950, vol. 183, no. 6, pp. 22–24, and the discussion of this article in *Scientific American*, February, 1951, vol. 184, no. 2, pp. 2–6.
An interesting and entertaining introduction to symbolic logic, and possibilities of its application. In the discussion some of the inaccuracies in the article are corrected. There is likely to be some disagreement with some of the more optimistic statements in this article.

LANGER, SUSANNE K., *An Introduction to Symbolic Logic*, Boston, Mass.: Houghton Mifflin Co., 1937, 363 pp; reprinted by Dover Publications, New York.
A good introduction with emphasis on philosophical aspects.

BLACK, MAX, *Critical Thinking: An Introduction to Logic and Scientific Method*, New York: Prentice Hall, Inc., 1946, 402 pp.
An interesting and useful exposition.

BERKELEY, EDMUND C., "Boolean Algebra (The Technique for Manipulating "And," "Or," "Not," and Conditions) and Applications to Insurance" in *Record of the American Institute of Actuaries*, vol. 26, part 2, Oct., 1937, pp 373–414; also the discussion by T. N. E. Greville, H. M. Sarason, and E. C. Berkeley, vol. 27, part 1, June, 1938, pp 167–176; reprinted by Berkeley Enterprises, Newtonville, Mass., 1952.
An application of Boolean algebra to the removal of conflicts and loopholes in actually occurring contracts and rules.

BERKELEY, EDMUND C., *A Summary of Symbolic Logic and its Practical Applications*, Newtonville, Mass: Berkeley Enterprises, 4th printing, 1957, 24 pp.
Rules for calculating with Boolean algebra; a brief description of other parts of symbolic logic; and applications of Boolean algebra to computing machinery, circuits, and contracts, with many complete problems and solutions.

BERKELEY, EDMUND C., *Symbolic Logic—Twenty Problems and Solutions*, Newtonville, Mass.: Berkeley Enterprises, 2nd printing, 1955, 28 pp.
Contains twenty complete problems and solutions in Boolean algebra and other parts of symbolic logic, some by Lewis Carroll and John Venn (out of print), and others new. Guide to using symbolic logic in actual situations.

TARKSI, ALFRED, *Introduction to Logic, and to the Methodology of the Deductive Sciences*, New York: Oxford University Press, 1941, 239 pp.
A first-class introduction to symbolic logic based more on mathematical examples than on philosophical examples.

WOODGER, J. H., *The Axiomatic Method in Biology*, Cambridge, England: The University Press, 1937, 174 pp.
Chapter 2, pp 18–52, is an excellent summary of the main concepts of mathematical logic, based on the ideas in *Principia Mathematica*.

CARNAP, RUDOLF, *Introduction to Symbolic Logic and Its Applications*, translated by Wm. H. Meyer and John Wilkinson, New York: Dover Publications, Inc., 1958, 241 pp.

QUINE, WILLARD VAN ORMAN, *Mathematical Logic*, New York: W. W. Norton & Co., Inc., 1940, 348 pp.

DAVIS, MARTIN, *Computability and Unsolvability*, New York: McGraw-Hill Book Co., Inc., 1958, 210 pp.

3. SYMBOLIC LOGIC AND SWITCHING CIRCUITS

KEISTER, WILLIAM, A. E. RITCHIE, and S. H. WASHBURN, *The Design of Switching Circuits*, New York: D. Van Nostrand Co., 1951, 576 pp.

BERKELEY, EDMUND C., *Circuit Algebra—Introduction,* Newtonville, Mass.: Berkeley Enterprises, revised printing 1953, 34 pp.

Boolean algebra modified to include time, that applies to on-off circuits, both static and sequential.

CALDWELL, SAMUEL H., *Switching Circuits and Logical Design*, New York: John Wiley & Sons, Inc., 1958, 686 pp.

4. LOGICAL DESIGN OF REASONING AND COMPUTING MACHINES

PHISTER, MONTGOMERY, JR., *Logical Design of Digital Computers*, New York: John Wiley & Sons, Inc., 1958, 408 pp.

BERKELEY, EDMUND C., *Giant Brains or Machines that Think*, New York: John Wiley & Sons, Inc., 1949, 270 pp.

Chapter 9 is devoted to a description of the Kalin-Burkhart Logical Truth Calculator.

BERKELEY, EDMUND C., and ROBERT A. JENSEN, *Constructing Electric Brains*, Newtonville, Mass.: Berkeley Enterprises, 3rd printing, 1957, 40 pp.

Reprint of a series of thirteen articles published in *Radio Electronics*, Oct. 1950 to Oct. 1951, which explained how an automatic computer is constructed; how to make it add, subtract, multiply, divide, and solve problems automatically.

BERKELEY, EDMUND C., *Brainiacs—Simple Electric Brain Machines and How to Make Them—Including Geniacs and Tyniacs*, Newtonville, Mass.: Berkeley Enterprises, 1955–58, 198 pp.

Collection of 8 reports describing problems and solutions, and giving circuit diagrams, for 151 Brainiacs, small electric brain machines that reason or compute arithmetically or logically, solve puzzles, play games (including Tit-Tat-Toe and Nim), test intelligence, encipher and decipher, etc. Included in the Brainiac K 17 kit for constructing small electric brain machines. Five of these Brainiacs are included in Chapters 8 and 9.

5. MISCELLANEOUS

THOMSON, SIR GEORGE PAGET, *The Foreseeable Future*, London, England: Cambridge University Press, 1955, 166 pp.

INDEX

C

D

E

T

SOME SIGNS