Reprinted from:
HANDBOOK OF PERCEPTION, VOL. I

Chapter 17

AUTOMATA

PATRICK SUPPES AND WILLIAM ROTTMAYER *

I. INTRODUCTION

The subject of this chapter is still in an embryonic stage and consequently the survey we present does not have a finished character. From many directions there is great current interest in the relation between perception and the theory of automata, because, increasingly, emphasis is on understanding how processing of energy as it impinges on the peripheral nervous system in the form of sensations is performed by the organism. From another standpoint, there is great interest in computer science in understanding how machines can be taught to perceive and to learn perceptual concepts.

The problem of providing a proper survey of the work in computer science relevant to the general theme of perception and automata is far from simple. It would be inappropriate in this handbook to give a detailed account of the work in computer science. On the other hand, a good part of the work we describe has originated in the context of computer efforts at pattern recognition or perception. Because the bibliography of this effort is now in itself substantial, we cannot hope to give the reader a really adequate account of what has been done.

To illustrate methods of approach we have given an undue emphasis to our own work, an emphasis not commensurate with its importance. We have used it in several places as an example to give the reader a sense of methods. We have also tried in the bibliography to give a number of leads

* Research connected with this article was partially supported by National Science Foundation Grant NSFGJ-443X.

into the literature, and we urge the reader to follow them up in order to get a detailed sense of the approaches that currently seem promising.

Since the bulk of the theoretical work on perception (including our own) has been on vision, we confine ourselves almost exclusively to remarks in this area. We begin by offering an abstract characterization of the perceptual process, which may be used to describe all of these approaches. Each approach is obtained by placing different restrictions on the abstract characterization, which allows one to pinpoint the essential features of each approach succinctly and to compare and contrast the differences between them.

We think of a perceptual device as being capable of telling whether what it perceives (a scene) has a certain property. Thus there are only two possible outputs: 1 (corresponding to a yes answer) and 0 (corresponding to a no answer). A device that can do this must be rather complex. Seven different features must be specified in order to define such a device mathematically:

1. $\mathcal{D}$, the form of the original perceptual data, e.g., the state of the retina; mathematically, $\mathcal{D}$ is a family of structures (sets together with relations on the set) rather than simple sets.

2. Φ, the preprocesser, which looks at elements of $\mathcal{D}$ and outputs (coded) information about them; Φ is fixed (no learning), and hopefully fairly simple.

3. $\mathcal{C}$, the set of possible outputs of Φ; alternatively, the possible inputs to Ω; coded information about elements of $\mathcal{D}$.

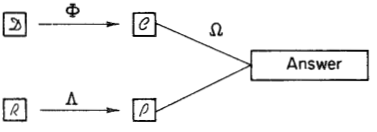
4. $\mathcal{R}$, the feedback (reinforcement); it may be a function of the answer.

5. Λ, the learning component, whose input is $\mathcal{R}$ and output is $\mathcal{P}$.

6. $\mathcal{P}$, the set of possible particular programs Ω can use to compute its answer; $\mathcal{P}$ is the output of Λ.

7. Ω, the processer, whose inputs are elements of $\mathcal{C}$ and $\mathcal{D}$, and which outputs the final answer; a general type of computing device that must be programmed to be able to handle any particular problem.

Schematically,*



Thus, Φ, Λ, and Ω are components of the computing devices, while $\mathcal{D}$ and $\mathcal{R}$ define the perceptual situation (environment). The question is al-

* Formally, Φ, Λ, and Ω are functions: $\Phi\colon \mathcal{D} \to \mathcal{C}$, $\Lambda\colon \mathcal{R} \to \mathcal{P}$ and $\Omega\colon \mathcal{C} \times \mathcal{P} \to \{0,1\}$ (1 is true or yes, 0 is false or no).

ways what particular devices can do in given situations. Thus the objects of study in the abstract are structures $\langle \mathcal{D}, \mathcal{R}, \mathcal{C}, \mathcal{P}, \Phi, \Lambda, \Omega \rangle$. In general, $\mathcal{C}$ and $\mathcal{P}$ are determined by the choices of the other five elements and are not of independent interest.

Let us agree that the ultimate goal is to define $\mathcal{D}$, $\mathcal{R}$, Φ, Λ, and Ω, so that the resulting overall system has roughly the same perceptual abilities as a human being. (For specific purposes, one might want to improve on human performances, e.g., make the receptors more sensitive or eliminate lapses of attention.) This goal is unlikely to be achieved immediately. Thus it seems wise to try to reach it by a series of steps, starting with systems that are extremely limited by human standards and trying to improve them. It is not a telling criticism of a particular system that it cannot match human performance, because every system proposed so far is deficient in some respect. Because this is a new area, it is natural to try to apply knowledge gained from other areas. Indeed, one way to distinguish different approaches is to specify what already known body of knowledge the approach is based on.

A satisfactory overall system would be able to do two things: (a) perform in a manner roughly equivalent to humans, and (b) learn in the way that humans do. Neither of these capabilities is presently simultaneously realizable. There are thus two approaches, each of which emphasizes one aspect and neglects the other. An artificial-intelligence approach emphasizes (a) and neglects (b), while a learning approach emphasizes (b) and neglects (a). Thus artificial-intelligence devices can perform tasks more complicated than learning devices, but learning devices are more adaptable and need less specific programming.

It is natural to ask questions that do not depend on the learning process about any approach. Suppose one has placed restrictions on Φ and Ω. What sorts of things can the resulting device do (given an appropriate element of $\mathcal{P}$) and what can it not do (given any element of $\mathcal{P}$)? The former question is about the *power* of the device, the latter about its *limitations*. These questions depend solely on $\mathcal{P}$, the whole range of possible programs for Ω. How particular elements of $\mathcal{P}$ are obtained, which is the whole of the learning question, is irrelevant. To put the matter a little differently, asymptotic learning studies (those that ignore practical questions like length of time and amount of memory) deal with whether Λ will eventually find a correct element of $\mathcal{P}$. They usually read "If there is at least one correct element in $\mathcal{P}$, then procedure Λ will eventually find a correct one." In order to apply such a theorem to a particular case, one needs to know that the antecedent clause is true. Conversely, if it is known that the antecedent clause is false then Λ cannot be successful, and wasted effort may be avoided. Questions about the truth or falsity of the antecedent clause are

questions about the power and limitations of the device. These are the questions of interest. The main part of this chapter summarizes three approaches to the problem, keeping these considerations in mind.

The perceptron approach discussed in Section II is a learning approach. It originated from considering how the brain might be organized to solve perceptual problems. Our work, discussed in Section III, is also a learning approach. It is based on stimulus-response learning theory, rather than models of the brain, however. A connection can be made (Suppes, 1969) between stimulus-response theory and automata, and we try to exploit this connection. The work in Section IV on picture-parsing illustrates an artificial intelligence rather than learning approach. In Sections II–IV we neglect learning and concentrate on questions of the power and limitations of the different approaches. Remember, however, that the choices of Φ and Ω made in Sections II and III were made to facilitate learning. Learning is discussed in Section V. We note that this characterization of the problem is based on the work of Minsky and Papert (1969) and Block's remarks on their work (Block, 1970), as well as on our own thinking.

In addition to the historical importance attached to visual perception, there are other reasons for concentrating on it in the present chapter. Perhaps the best is that our understanding of the geometry of visual perception is very much better than the geometry of any of the other senses. With the single exception of the auditory processing of language, almost all the applications of automata to perception have been restricted to visual perception. It is perhaps worth pointing out the theory we develop in Section III can be applied almost without change to tactile perception. We take the notions of straight line and intersection as primitive. Consequently, given a drawing with raised lines, for example, a Braille drawing, a person can gather the information necessary for perception by tactile methods. The fact that this would require moving the fingers and hence would take time is not essential, since what is required is that the device at some time have all the information at its disposal, not that it be gathered all at once. Of course, hypotheses about motion are always an important part of visual perception as well. We shall not pursue this tactile theory, but it will be obvious to the reader that the technical developments in Section III apply without serious change to such tactile perceptions.

The remainder of this section is devoted to some general remarks about automata. The first automata of which we have precise details and of which we have any serious historical information appeared in the second or third century B.C. in Alexandria. Perhaps the most striking example is the mobile theatre of Hero of Alexandria, which had a large number of moving parts, including wine flowing from the statue of Bacchus and the sounds of drums and cymbals. The ancient literature also gives a fairly detailed description of how the mobile theatre was built and of its mechanical principles of

construction. The more practical applications were to hydraulic clocks or other forms of water clocks. Needham (1965) has also emphasized the independent development of mechanical toys, clocks, and other devices in China.

From the fifteenth century on there are numerous examples of mechanical automata doing a surprisingly wide variety of things, ranging from clocks to toys and mechanical conjurers. Many subtle functions are performed by these mechanical automata and from a surprisingly early date. What is missing by and large, however, is any sort of development bearing on the central concern here, that of perception. In the few cases where a mechanical automaton depended upon some kind of input, the input was too simple to treat it as a case of perception. To take a modern instance, an elevator that keeps its door open as long as a light beam is interrupted more often than every 10 or 15 seconds is an example of a very simple automaton depending upon an extremely simple visual input, but we would not want to treat it as a case of perception. It is not our point here to draw a sharp distinction. We do want to emphasize that in the long history of mechanical automata, including even efforts at talking automata in the eighteenth century, problems of perception scarcely entered. For example, in 1779, the Imperial Academy of Science of St. Petersburg set as the subject for its annual prize investigation into the nature of sounds, in particular of the vowels, and the building of a device capable of reproducing them. The prize was won by Kratzenstein, who used a bellows that forced air into tubes of different shapes. Methods of this kind were extended to some words and phrases, but no conceptually interesting results were obtained. Even in the twentieth-century development of mechanical robots prior to the introduction of computers in the late 1940's, little attention was paid to perception. The theoretical ideas developed in this chapter and the literature referred to make clear why this was the case. The subject is difficult, and progress is even now relatively slow.

Although, as indicated, most of the current conceptual work that is specifically oriented toward perception is concerned with visual perception, the work on generative grammars in the late 1950's by Chomsky and others provides one close link between automata and perception, in this case the perception of spoken speech. Much of the formal work has not been concerned with the details of phoneme perception or with the more elementary distinctive features of phonemes, but it has been implicitly assumed that the distinctive features of the "meaningful" elements, for example phonemes, of spoken speech are finite in number, and a theory of language built up from a finite vocabulary has been extensively developed. A good introduction to these matters is to be found in Chomsky and Miller (1963).

Assuming without detailed justification that speech can be broken down

into a sequence of discrete elements made up from a finite set of such elements, we can then ask what kinds of abstract machines are required to process speech of different levels of complexity. Put in this abstract way the problem may be seen as far removed from perception, but as we shall see in several different contexts in this chapter, once the restriction to a finite set of basic elements is made, a good many subtle distinctions can be introduced and certain theoretically interesting questions can be answered.

We turn now to a brief sketch of the general theory of automata and the languages a given class of automata can recognize or accept. We begin with the definition of a finite automaton which is close to the definition originally given by Rabin and Scott (1959).

DEFINITION 1. A structure $\mathfrak{A} = \langle A, V, M, s_0, F \rangle$ *is a finite (deterministic) automaton iff**

(i) A *is a finite, nonempty set (the set of states of* $\mathfrak{A}$),

(ii) V *is a finite, nonempty set (the alphabet or vocabulary)*,

(iii) M *is a function from the Cartesian product* $A \times V$ *to* A *(M defines the transition table of* $\mathfrak{A}$),

(iv) s_0 *is in* A *(s_0 is the initial state of* $\mathfrak{A}$),

(v) F *is a subset of* A *(F is the set of final states of* $\mathfrak{A}$).

The generality of this definition is apparent. It is also indicative of the weakness in a certain sense of the general notion of a finite automaton. About the only restrictions are that the sets A and V be finite. On the other hand, these finite restrictions are critical.

It is evident that we can have some extremely trivial models of Definition 1. Here is the simplest: $A = \{s_0\}$, $V = \{0\}$, $M(s_0,0) = s_0$, $F = \phi$, where ϕ is the empty set (also later the empty sequence).

Perhaps the simplest nontrivial example of a finite automaton is the following. We have a two-letter alphabet consisting of the symbols "0" and "1," and two internal states, s_0 and s_1. The transition function of the automaton is defined by the following tabulation:

	0	1
s_0	✓	
s_1		✓
s_0		✓
s_1	✓	

* Throughout, "iff" is used as an abbreviation for "if and only if."

Finally, we select the internal state s_1 as the only member of the set F of final states. In saying that this is the simplest nontrivial automaton, we mean that the transition table depends both on the internal state and the input letter. From a more general conceptual standpoint, it is clear that the device is itself pretty trivial.

We next describe the language that a given finite automaton can accept or recognize. First, V^* is the set of all finite sequences of elements of V, including the empty sequence ϕ. The elements of V^* are called *sentences, strings,* or *tapes.* If $\sigma_1, \ldots, \sigma_n$ are in V, then the sequence $\sigma_1\sigma_2 \ldots \sigma_n$ is in V^*; in other words, we shall usually show elements of V^* by juxtaposing names of elements of V.

Second, the function M can be extended to a function from $A \times V^*$ to A by the following recursive definition for s in A, x in V^* and σ in V

$$\begin{aligned} M(s,\phi) &= s \\ M(s,x\sigma) &= M(M(s,x),\sigma). \end{aligned}$$

Thus, for the two-state, two-letter-alphabet automaton introduced above, we may compute M for the string $x = 101$ by using this recursive definition.

$$\begin{aligned} M(s_0,101) &= M(M(s_0,10),1) \\ &= M(M(M(s_0,1),0),1) \\ &= M(M(s_1,0),1) \\ &= M(s_1,1) \\ &= s_0, \end{aligned}$$

and so the string 101 is not accepted, because the final state is s_0, not s_1, the only member of F. More formally, we have

DEFINITION 2. *A string x of V^* is accepted by $\mathfrak{A}$ iff $M(s_0,x)$ is in F.*

It is also customary to call strings that are accepted by $\mathfrak{A}$, *sentences* of $\mathfrak{A}$.

DEFINITION 3. *The language accepted by $\mathfrak{A}$ is the set of all sentences of $\mathfrak{A}$.*

In the literature, the language accepted by $\mathfrak{A}$, which we shall denote $L(\mathfrak{A})$, is often called the *set of tapes accepted by* $\mathfrak{A}$.

There is a natural notion of equivalence for automata which is weaker than the natural isomorphism of states and transitions for given inputs.

DEFINITION 4. *Two automata are (weakly) equivalent iff they accept the same language.*

In order to talk about the languages accepted by automata as defined independent of any machine concepts, we must introduce the general concept of a *grammar.*

Let V be a finite, nonempty set. Then V^* is the set of all finite sequences

of elements of V. Let ϕ be the empty sequence. Then $V^+ = V^* - \{\phi\}$. V is the *vocabulary*. V^* is the set of *strings* or *tapes*.

DEFINITION 5. *A structure* $G = (V,N,P,S)$ *is a phrase-structure grammar iff*

(i) V, N, *and* P *are nonempty finite sets,*
(ii) $N \subseteq V$,
(iii) $P \subseteq V^+ \times V^*$,
(iv) $S \in N$.

The standard terminology is this. The set N is the set of *nonterminal* symbols, or variables. $V_T = V - N$ is the set of *terminal* symbols. The set P is the set of *productions*. If $\langle \alpha,\beta \rangle \in P$ we ordinarily write: $\alpha \rightarrow \beta$. Finally, S is the *start* symbol.

DEFINITION 6. *If* γ *and* δ *are in* V^* *then* $\gamma\alpha\delta \Rightarrow {}^G\gamma\beta\delta$ *iff* $\alpha \rightarrow \beta$ *is a production, i.e.,* $\langle \alpha, \beta \rangle \in P$.

DEFINITION 7. *Derivation of* γ_m *from* γ_1, *with* $\gamma_1,\gamma_m \in V^*$. $\gamma^1 \Rightarrow_G^* \gamma_m$ *iff* $\exists\ \gamma_2, \ldots, \gamma_{m-1} \in V^*$ *such that*

$$\gamma_1 \Rightarrow_G \gamma_2, \ldots, \gamma_{m-1} \Rightarrow_G \gamma_m.$$

DEFINITION 8. *The language generated by a grammar* G, *denoted* $L(G)$, *is:*

$$L(G) = \{w | w \in V_T^* \ \& \ S \Rightarrow_G^* w\}.$$

DEFINITION 9. *Two grammars* G_1 *and* G_2 *are (weakly) equivalent iff* $L(G_1) = L(G_2)$.

Example 1

As an example, here is a tiny subfragment of English, for which we make no claims of empirical correctness.*

$N = \{\text{S,NP,VP,PN,Adj P,Adj,N,Art,V}_1\text{,V}_2\}$
$V_T =$ a small set of English words
$P =$ the set of following production rules

1. $\text{S} \rightarrow \text{NP} + \text{VP}$
2. $\text{NP} \rightarrow \text{PN}$
3. $\text{NP} \rightarrow \text{Adj P} + \text{N}$
4. $\text{Adj P} \rightarrow \text{Adj P} + \text{Adj}$
5. $\text{Adj P} \rightarrow \text{Adj}$
6. $\text{Adj P} \rightarrow \text{Art}$
7. $\text{VP} \rightarrow \text{V}_1$
8. $\text{VP} \rightarrow \text{V}_2 + \text{NP}$

* The nonterminal symbols have an intuitive meaning taken over from classical grammars: S for sentence, NP for noun phrase, VP for verb phrase, PN for proper noun, AdjP for adjective phrase, Adj for adjective, N for noun, Art for article, V_1 for intransitive verb, and V_2 for transitive verb.

Here is a sample derivation:

1.	S	Axiom
2.	NP + VP	By Rule 1
3.	Adj P + N + VP	By Rule 3
4.	Adj P + Adj + N + VP	By Rule 4
5.	Art + Adj + N + VP	By Rule 6
6.	Art + Adj + N + V_2 + NP	By Rule 8
7.	Art + Adj + N + V_2 + PN	By Rule 2
8.	The large ball hit John	By Lexical Rules

We turn now to types of grammar. *Type* 0 *Grammars* are characterized by Definition 5. To obtain a *Type* 1 or *Context-Sensitive Grammar*, we add the restriction that for every production $\alpha \to \beta$, $|\alpha| \leq |\beta|$, where $|\alpha|$ is the length of α, or the number of symbols in α. This restriction can be shown equivalent to

$$\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$$

with $\beta \neq \phi$ and $A \in$ N.

To obtain *Type* 2 or *Context-Free Grammars*, we add the stronger restriction: if $\alpha \to \beta \in P$ then

(i) α is a variable, i.e., $\alpha \in N$,
(ii) $\beta \neq \phi$.

To obtain *Type* 3 or *Regular Grammars*, we add the still stronger restriction: any production must be of the form

(i) $A \to aB$
or, (ii) $A \to a$,

with $A,B \in N$, $a \in V_T$. (We can have instead of (i), $A \to Ba$.)

For an example of a regular grammar, we may slightly change our previous example.

Example 2

$$N = \{\text{S,NP,VP,Adj P}\}$$
$$V_T = \{\text{PN,N,Adj,Art,}V_1,V_2\}.$$

If we drop Rule 8 and rewrite Rule 1 as 1′, $\text{S} \to \text{NP} + V_1$, we can immediately check that the grammar is regular. To rewrite Rule 8 is more troublesome and requires more variables, i.e., elements of N. We leave the details to the reader. In general, it is not easy to see if a *language* has a grammar of a given type.

We turn now to relations between languages and automata.

There are four basic theorems about the types of automata that can recognize or process the four types of languages, Types 0–3. It is too far from our main concern to give a detailed treatment of these matters, but we will use them later in discussing picture-parsing grammars and in remarks on coding for automata. Consequently we review the classical results briefly. The reader is referred to Chomsky and Miller (1963) and Chomsky (1963), or to the excellent recent textbook Hopcroft and Ullman (1969).

The first basic theorem is that a language is recognized by a finite automaton if and only if it has a regular grammar. Intuitively this means that the structure of the language is simple. A critical property for example is the absence of imbeddings.

The next basic theorem for context-free languages requires the notion of a pushdown automaton, and we describe the character of such automata in an informal way. The new idea introduced is that in addition to having a finite number of control states, as in the case of the finite automaton, the pushdown automaton has a pushdown store, which is a restricted form of memory. The automaton can put structural information in the store, but it can make state transitions only in terms of the top symbol on the store—the store operates like a stack of cafeteria trays on the principle of "first in and last out." In addition, a special store-vocabulary is provided the machine and the transition function depends not just on the input symbol and the current state as in the case of the finite automaton, but also on the top symbol of the store. The basic theorem is that a language can be recognized or processed by a pushdown automaton if and only if it has a context-free grammar.

For the consideration of context-sensitive and Type 0 languages we need to introduce the concept of a Turing machine. A Turing machine has a single read-write head. Given an internal state and the scan of an input letter, the Turing machine executes a move by changing its internal state, printing a nonblank symbol on the cell scanned (thereby replacing the symbol scanned), and moving its head left or right one cell. As in the case of a finite automaton the set of internal states is finite as is the input alphabet. A language is of Type 0, that is, has a Type 0 grammar, if and only if it is accepted by some Turing machine. The general class of Type 0 languages is so large that they are themselves of little interest. For example, there are Type 0 languages that are recursively enumerable but not recursive. This means that there is no mechanical decision procedure for deciding whether or not a string made up of the terminal vocabulary is or is not a sentence of the language. It is generally felt that languages are always recursive and consequently Type 0 languages form too large a class. We emphasize that problems of vagueness and unclarity about the grammar

of a language are quite separate from the question of whether or not it is recursive.

A linear bounded automaton is a Turing machine that stays within the squares of the tape on which the input is placed. The basic theorem for context-sensitive languages is that a language is context-sensitive if and only if it is accepted by some linear bounded automaton. With reference to the remark just made about recursion, it should also be noted that any context-sensitive language, that is, any language that has a context-sensitive grammar, is recursive.

II. PERCEPTRONS

In this section we summarize the work of Minsky and Papert (1969), but also use the review of their work in Block (1970). Earlier relevant work is summarized in Rosenblatt (1959, 1962). The power and limitations of perceptrons are presented, with the discussion of learning being postponed until Section V. We begin with some brief historical remarks, then define perceptrons and the perceptual situation dealt with by Minsky and Papert, give their main results, and conclude with comments on the significance of their results.

The perceptron, as a relatively simple linear device, has had some surprising successes, notably Samuel's checker player (Samuel, 1959, 1967). These successes, plus the perceptron's appealing similarity to known physiological structures, led to the widespread belief that it was a satisfactory model for the perceptual process. Although no concrete model that has abilities even remotely resembling human capabilities has ever been produced, this belief has persisted. Minsky and Papert felt that the perceptron was not an adequate model of the perceptual process, and that its successes were limited to applications where a linear weighing of evidence was appropriate. Further, they felt that even in these cases the evidence to be weighed had to be judiciously chosen to make a correct overall solution feasible. Since many of their colleagues were unconvinced, they set out to prove their conjecture. It is these results that we summarize, and we begin with the definition of a perceptron.

The perceptron approach can be defined generally (without confining it to a particular context) by placing the appropriate restrictions on the preprocessor Φ and the processor Ω. The basic idea is that Φ computes the answers to many simple questions, and Ω combines these answers linearly to answer a more complex question. There are two big attractions (besides its similarity to physiological structures) to this approach: Ω is simple (so

learning can be studied easily) and Φ operates in parallel (computing many things at once). The preprocessor Φ answers a question by stating whether a statement, $P(X)$, is true of X. Formally, P can be represented as a function from all the possible values of X into $\{0,1\}$ with 0 corresponding to P being false, and 1 to P being true. Since Φ operates in parallel, it can compute more than one predicate P at a time. Hence, the predicates must be calculable independently. This means that Φ must consist of a set of predicates, rather than one predicate. Let φ's henceforth represent typical elements of Φ. This definition of Φ determines the nature of the output of Φ, which must consist of n-tuples of 0's and 1's. The processor Ω takes as its inputs the n-tuples of 0's and 1's ($\langle X_1, X_2, \ldots, X_n\rangle$) and $(n+1)$-tuples of real numbers ($\langle \alpha_1, \alpha_2, \ldots, \alpha_n, \theta\rangle$, where the α_i's are coefficients and θ is the threshold) and is defined by

$$\Omega(\langle X_1, X_2, \ldots, X_n\rangle, \langle \alpha_1, \alpha_2, \ldots, \alpha_n, \theta\rangle) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} \alpha_i X_i > \theta, \\ 0 & \text{otherwise.} \end{cases}$$

A predicate ψ is said to be a *linear threshold function* with respect to Φ if there are coefficients α_i and a threshold θ such that $\Omega = 1$ if ψ is true, and $\Omega = 0$ if ψ is false. The set of all linear threshold functions with respect to Φ is denoted by $L(\Phi)$. The coefficients and threshold come from the set $\mathcal{P}$ of "programs"; hence, $\mathcal{P}$ is the set of all $(n+1)$-tuples of real numbers. How particular elements of $\mathcal{P}$ are selected to make a specific calculation is a learning question and is discussed in Section V.

The general definition of perceptrons, as stated formally, is uninteresting. Because there are no restrictions on the predicates in Φ, it is possible to choose a complex partial predicate that "does all the work" and, hence, trivializes Ω. (For any ψ, simply make ψ an element of Φ, and Ω the identity function.) What is needed to capture the intuition behind perceptrons is to restrict the partial predicates to being "simple" functions. This cannot be done with complete generality since there is no general theory for the simplicity of functions. However, in particular contexts it is possible to give satisfactory ad hoc definitions of simplicity. Technically speaking, the data $\mathcal{D}$ must be specified before a precise notion of simplicity can be given.

Minsky and Papert use two different definitions of $\mathcal{D}$, which gives rise to two different theories: an algebraic theory and a geometric one. Although the results of the geometric theory are of primary interest to us, we mention certain general concepts that are best defined in terms of the algebraic theory.

Let $\mathcal{D}$ be the class of all finite sets. (Minsky and Papert take $\mathcal{D}$ to be the class of all sets, but their main results are proved for families of finite sets.)

An arbitrary element of $\mathfrak{D}$ is denoted by R and is called a retina. Thus R denotes an arbitrary finite set with no structure. The domains of the φ's, the predicates in Φ, are subsets of R. Since the domains are not in general all of R, the φ's are called *partial predicates*. The complex predicate, ψ, computed by Ω, has as domain all of R. To specify ψ, it is necessary to specify R. Normally, we are not interested in predicates tied to a certain R, but rather in predicate schemes defined for a family $\mathfrak{D}$ of R's, and we shall let ψ also represent a predicate scheme. Formally, ψ is a function from $\mathfrak{D}$ into the set of predicates defined on elements of $\mathfrak{D}$, such that $\psi(\mathrm{R})$ is some predicate defined on R.

We are now in a position to define an appropriate definition of simplicity for predicate schemes, but this requires two preliminary definitions.

DEFINITION 10. *For any element φ of Φ, $S(\varphi)$, called the support of φ, is the smallest subset S of* R *such that for every subset X of* R, $\varphi(X) = \varphi(X \cap S)$.

Intuitively, the support of φ is the subset of R on which φ "really depends." As examples let $\psi(X)$ hold iff X is not empty. Then the support of ψ is the whole retina R. In contrast, if $\psi(X)$ holds iff $\{p_1,p_2\}$ is in X, then $\{p_1,p_2\}$ is the support of ψ.

DEFINITION 11. *The order of a predicate ψ is the smallest number k, for which there is a set of predicates Φ such that $|S(\varphi)| \leq k$ for all φ in Φ* and $\psi \in L(\Phi)$.

Using the first of the examples of support, where R is the support of ψ, the order of ψ in contrast is 1, because we take as Φ the set of completely local predicates φ_p: $\varphi_p(X)$ holds iff p is in X. Then the support of any φ_p is just $\{p\}$ and $|S(\varphi)| = 1$, and $\Psi(X) = \Sigma\varphi_p(X) > 0$.

DEFINITION 12. *A predicate scheme ψ is of finite order, in fact of order $\leq k$, on a family of sets $\mathfrak{D}$ if for every element* R *of D, $\psi(\mathrm{R})$ has order $\leq k$.*

Henceforth, we are no longer interested in predicates (only predicate schemes), so we follow Minsky and Papert and call predicate schemes predicates. All the results mentioned below about predicates are really about predicate schemes. The last definition, then, is the operative one. The smaller k is, the simpler the predicate (scheme). This is the intuitive idea. The important use of the definition is to rule out predicates *not* of finite order as being calculable by a perceptron. This would follow if there were some limit on the supports of the elements of Φ, a restriction which seems reasonable and gives the theorems more than just technical interest.

In the geometric theory an element R of $\mathfrak{D}$ is obtained by dividing a rectangular region of the Euclidean plane into equal squares. The set of

squares is R, and the set of all such R's is $\mathfrak{D}$. For a subset Y of a rectangular region, a corresponding subset in an appropriate R is given by x is in X if at least one point of Y lies in the square x. For example, X is a circle if it can be obtained from a real circle Y by this rule. However, apparent errors may occur near the "limits of resolution" because of this; e.g., small circles will not look very round, and serious problems may result when dealing with real figures. Minsky and Papert avoid these problems by starting with figures defined on R and then deal only with translations which are a multiple of the length of the sides of a square and 90 degree rotations.

Two squares in R are *adjacent* if they have a common edge. A subset X of R is *connected* if for every two squares in X there is a path of adjacent squares connecting them.

THEOREM 1. *The predicate $\psi(X)$ iff X is connected is not of finite order. In fact, it is of order $\geqq C|\mathrm{R}|^{\frac{1}{2}}$.*

This is one of the most significant theorems in the book of Minsky and Papert. The algebraic theory was developed as a tool for proving it. Minsky and Papert felt that the partial predicates must be in some sense "local," and that it would prove impossible to compute "global" functions on the basis of these "local" ones. Their paradigm of a global feature was "connectedness." Recognition of connectedness by a different device is discussed in Section III.

A *component* of a figure is the set of all squares connected to a given square. A *hole* of a figure is a component of the complement (where squares touching only at a corner are considered to be adjacent). Every figure is assumed to be surrounded by an outside that does not count as a hole. A predicate is *topologically invariant* if it remains unchanged whenever the figure is distorted without changing connectedness or inside-outside relations of its parts.

THEOREM 2. *Let the Euler number $E(X) = |\ components(X)\ | - holes\ |X|$. Then the only topologically invariant predicates of finite order are functions of the Euler number.*

The above results are negative. The following results are positive.

We first mention some familiar geometrical properties that have low finite order.

THEOREM 3. *The following predicates are of order* 3: *convexity, solid rectangles, and hollow rectangles.*

THEOREM 4. *Except for resolution problems, the predicate that X is the perimeter of a complete circle is of order* 4.

The following results depend on "stratification," a technique which simu-

lates serial computation in the parallel framework. Stratification requires such large coefficients that it is not achievable in a practical way, but it is of theoretical interest.

THEOREM 5. *Let* R *consist of a line of squares, infinite in both directions. Then the predicate that X has a symmetry under reflection is of order* ≤ 4 *for all finite X.*

THEOREM 6. *Let* R *consist of two infinite lines of squares A and B. Then the predicate that the (finite) figure in A is a translate of the figure in B is of order* ≤ 5.

That context can be crucial is illustrated by the following pair of theorems.

THEOREM 7. *The predicates that X is a solid square and X is a hollow square are of order* ≤ 3.

THEOREM 8. *The predicate that X is a hollow square in context is not of finite order.*

In general, it seems that most of the predicates "in context" are not of finite order, although this is proved only for a few predicates. Insofar as this is the case, it is a serious limitation on perceptrons.

Without casting doubt on the ingenuity and importance of Minsky's and Papert's work, for our purposes the significant issue is the continuing potentiality of the perceptron approach to perception. As Block (1970) points out, Minsky and Papert have defined perceptrons in such a way that only the simplest type of perceptron may be called a perceptron. These simple perceptrons have received a great deal of attention, but they were accorded this interest only because they were seen as the simplest type in a series of progressively more complicated models. It seemed wise to study the simplest case thoroughly before proceeding to the more complicated models, which are the objects of real interest. Minsky and Papert probably do not do justice to the view that the simple perceptron is only of temporary interest.

In addition, their geometric results are tied to their definition of $\mathfrak{D}$, i.e., the type of retinas they deal with. It seems plausible that all their results (with minor modifications) would hold if one divided the plane into, e.g., hexagons, rather than squares. However, there are other choices for $\mathfrak{D}$, e.g., line drawings, for which it is not so obvious that the same results would hold.

In conclusion, it seems that Minsky and Papert have not established that the perceptron approach should be abandoned in trying to model the perceptual process. They have shown that there are good reasons for following

other approaches. We turn to some relatively simple alternatives in the next two sections.

III. AUTOMATA AND LINE DRAWINGS

The intuitive idea behind the approach discussed in this section is to keep the preprocessor Φ simple and to have the processor Ω act like an automaton (not necessarily a finite one). An automaton acts "serially" rather than in "parallel" and, hence, differs considerably from the perceptrons of the previous section.

An automaton takes as its input a string of symbols that has on the surface no natural geometric properties. In order to do geometry with an automaton, it is necessary to code a figure with geometric properties so that it is an appropriate input for an automaton. There is no straightforward way to code all the usual geometric properties into a string of symbols. The solution to this problem is to choose some natural coding and then to investigate which geometric properties can and which cannot be recognized.

The above considerations suggest the following choices. Let $\mathfrak{D}$ consist of only straight-line drawings, which are henceforth called *figures*. For each figure in $\mathfrak{D}$, label the vertices with capital roman letters, followed when necessary by primes. Each line in the figure is represented by an n-tuple of labels. A tape for the figure can then be obtained by listing the n-tuples, in any order, and separating them with a comma. As an example, take Fig. 1. Label the vertices, using any order. Thus, a tape for the figure is ABCDE, AG,BF,GCH,HD.

Any permutation of the labels or rearrangement of the order in which the lines are listed will also yield a tape for the figure. This method of coding figures requires a minor change in the way elements of the vocabulary V are written. We now use a comma, a prime symbol and the capital roman letters, e.g., A, B, . . . , A′, B′, . . . are labels. We list conditions that elements of V^*, the set of finite sequences of elements of the

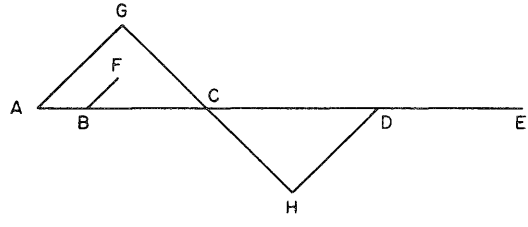


FIG. 1

vocabulary V, must satisfy to be codings of figures, with the least interesting first (and we omit obvious conditions on the labels themselves as letters followed by a possibly empty sequence of primes).

1. Commas must not appear at the beginning or end of the tape, nor may two commas follow each other.

2. Every sequence of labels coding a line segment of a figure must contain at least two distinct labels, but no label may occur twice.

3. There must be no "geometric" impossibilities. There is no complete characterization of these impossibilities, but they seem to be obtainable from a real drawing by deleting one vertex, e.g., ABC,ADE,BE,DC which is a coding for Fig. 2 with the vertex F omitted.

Intuitively, we have a straightforward method for coding figures as input tapes for automata, including Turing machines. Whatever sort of computing device is used, it will be limited by the limitations of its input.

The results fall into two categories: negative (what geometric information cannot be retrieved from a coding) and positive (what can be retrieved). The positive results are listed first, without proof, since the proofs are complex. The negative results are susceptible to shorter proofs, so we either give or indicate the main ideas in these proofs.

A coding is *connected* if there is a sequence $\langle b_0,b_1, \ldots ,b_n\rangle$ for every pair of lines c,d such that $b_0 = c$, $b_n = d$ and for all i, $1 < i < n$, b_i has a vertex in common with b_{i-1} and b_{i+1}. A *component* of a coding is a maximal connected subset of the coding.

Note that a coding is connected if a figure for which it is the coding is connected.

THEOREM 9. *There is an effective procedure (thus a Turing machine) for listing the components of a coding and, hence, for recognizing connectedness.*

We say that AB is a *segment* in a coding if there is a line $b = q\mathrm{A}r\mathrm{B}t$ in the coding, where q, r, and t may be either empty or nonempty. An n-tuple

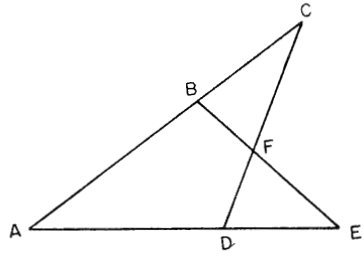


FIG. 2

$V_1V_2 \cdots V_n$ of the vertices which occur in a coding is a *polygon* if $V_1V_2, V_2V_3, \ldots, V_{n-1}V_n, V_nV_1$ are segments in the coding.

Note that if an n-tuple is a polygon in a coding, then the elements of the n-tuple list, in order, the vertices of a polygon in any figure which the coding is a coding of.

THEOREM 10. *There is an effective procedure (thus a Turing machine) for listing all the polygons in a coding.*

The procedures mentioned above are "reasonable," i.e., they do not require simply running through all possibilities. However, they require the capabilities of a Turing machine, rather than that of a finite automaton. The ability to look over the same n-tuple more than once becomes necessary, because no particular order has been specified for the elements on the tape. With suitable arrangement of lines, a pushdown automaton, for example, can recognize whether a given n-tuple of vertices is a polygon. How far this reduction of computing capabilites can be carried without requiring a particular arrangement of the elements on the tape is a topic for further investigation.

We now turn to some negative results.

THEOREM 11. *It is not always possible to tell whether a polygon in a figure is concave or convex given only a coding for the figure.*

Proof

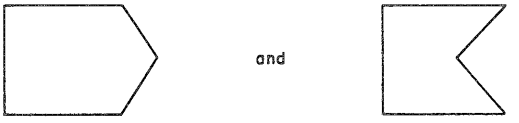
Figures 3A and B have the same, or equivalent, codings.

THEOREM 12. *It is not always possible to tell what line segments are inside (outside) of a closed figure (which must be a polygon) given only a coding for the figure.*

Proof

Figures 4A and B have the same coding.

THEOREM 13. *It is not always possible to recognize the simple regions in a figure [areas enclosed by a polygon but not crossed by any (broken) line], given only a coding for the figure.*



FIGS. 3A and B

FIGS. 4A and B

Proof

Figures 5A and B, each of which contains two hexagons and a quadrilateral, have the same coding.
In the first, the two hexagons are the simple regions, in the second the quadrilateral and a hexagon.

The above results are the rule, not the exception. Indeed, they hold for almost all relatively complex figures. Combined, the three results indicate that most topological properties are lost by this method of coding.

Additions can be made to the coding to alleviate some of the above difficulties and, hence, make the resulting tapes more closely resemble ordinary geometry. These possibilities shall not be discussed here.

IV. PICTURE-PARSING GRAMMARS

In the preceding two sections we have been concerned with automata or automatonlike devices that begin by observing local properties of a figure and build up, either from parallel or serial processing, the computations for a complex predicate. Another important approach that has developed in the past decade is to look at the more global characteristics of figures to see whether a grammar can be written for figures that is similar to the kind of grammars that have been written for languages. The connection with automata is then made directly by the theorems relating grammars and automata as outlined in the introductory section.

Because of the extensive interest in automatic pattern recognition by computers, the literature on picture-parsing grammars, or grammatical description of pictures, is now enormous, and it is not possible to review even the most important articles that have been published. Most of the

FIGS. 5A and B

published work has appeared since 1960. The early work of Eden (1961, 1962) and Narasimhan (1962, 1964, 1966) has been influential in subsequent developments. An important step in making the connection with grammars completely explicit and formal was taken by Shaw (1969), and we shall mainly use some of his examples to illustrate the theory.

How a picture language is intended to work can be illustrated by considering a simple grammar for generating those letters of the alphabet that are made up only of line segments. We shall follow with slight modifications an example used by Shaw (1969). The terminal vocabulary, V_T, consists of h, v, l, r, four operation symbols $+$, $-$, $\times$, and $*$, and parentheses. The meaning of h, v, l, and r is that they are segments of given length, with h being horizontal; v, vertical; l, a left diagonal; and r, a right diagonal. To interpret the operation symbols a direction is also given each segment, so that we can refer to the head and tail of each of the four primitive elements as shown in Fig. 6. The meaning of the operation symbols is reflected in

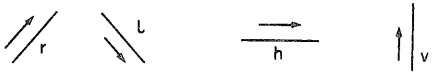


FIG. 6. Primitive elements.

just the four possible concatenations of heads and tails. For example, letting $\frown$ stand for the operation of concatenation,

$r + l = \text{head}\ (r) \frown \text{tail}\ (l)$ ⋀
$l + r = \text{head}\ (l) \frown \text{tail}\ (r)$ ⋁
$h - v = \text{head}\ (h) \frown \text{head}\ (v)$ ⌐
$h \times v = \text{tail}\ (h) \frown \text{tail}\ (v)$ ⌊
$r * l = \text{tail}\ (r) \frown \text{tail}\ (l)$ & $\text{head}\ (r) \frown \text{head}\ (l)$.*

Also, the head and tail of $S_1 0 S_2$ for any of the four operations 0 is the head of S_2 and the tail of S_1.

The picture grammar is simply stated in terms of these operations, requiring in this version only the single nonterminal symbol S. There are eight rules of production

$S \rightarrow (S + S)$
$S \rightarrow (S - S)$
$S \rightarrow (S \times S)$
$S \rightarrow (S * S)$
$S \rightarrow$ atom, where r, l, h and v are atoms.

* The operations are not always well defined for all primitive elements. Here $r * l$ is not defined, but $r * r = r$, for example, and another example is given in text.

The capital letter A has the following derivation, which of course is not unique:

S
$(S + S)$
$(r + S)$
$(r + (S + S))$
$(r + (S + l))$
$(r + ((S * S) + l))$
$(r + ((S * h) + l))$
$(r + (((S + S) * h) + l))$
$(r + (((r + l) * h) + l))$.

The terminal expression for the letter F is $(v + (h \times (v + h)))$, and for the letter L is $(h \times v)$; other examples are easily constructed.

The selection of primitive elements for such picture grammars could be guided by knowledge of the special types of receptor cells in the retina or other sensory organs, but this direction has not been pursued very far as yet. The main thrust, as indicated, has been toward pattern recognition and related problems in "perceptual" uses of computers. The theoretical and technical developments are now substantial, and it will be surprising if some of the methods and results do not prove highly useful in the study of perception by living organisms. Some steps in this direction are outlined in Barlow, Narasimhan, and Rosenfeld (1972).

The kind of grammar just exhibited has a number of defects when applied to pictures.

1. A good case can be made for parallel processing rather than sequential processing as required by a pushdown automaton for a context-free grammar. In a parallel grammar every instance of the left member is replaced by the right member of the production rule. The power of such grammars is essentially the same as that of the phrase-structure grammars defined in Section I; this is proved in Rosenfeld (1971). Use of such parallel grammars could lead to perceptually more appropriate picture grammars.

2. The grammar given above is context-free, but there is a variety of evidence that context-sensitive rather than context-free grammars are more useful for analysis of pictures. For example, using h and v as given above we can generate the set of all squares with horizontal base and n units per side, but even the simple language for this set is context-sensitive. The tendency in linguistics, on the other hand, is to restrict the base grammar to being context-free and then to apply transformations to this base. Increasing divergence between grammars for pictures and grammars for linguistic utterances seems likely in the future.

3. Most of the picture grammars as yet developed apply to line drawings and seldom if ever to pictures with continuous gradients of shading and color over the entire surface. How well the present framework can be successfully modified to analyze such "richer" scenes is not yet clear.

V. LEARNING

This section is divided into three parts. The first gives an abstract characterization of a learning situation, the second deals with perceptron learning, and the third with automaton learning.

In the general situation, learning takes place in a series of trials. Each trial is defined by (a) the state of conditioning at the beginning of the trial, (b) the presented object (stimulus), (c) the sampled stimuli, (d) the response given, (e) the reinforcement received, and (f) the new state of conditioning at the beginning of the new trial. Item (f) actually belongs to the next trial, and often the first five elements will not all be distinct. For example, sometimes no distinction may be drawn between the presented and sampled stimuli. A learning procedure is specified by saying how the state of conditioning changes from one trial to the next, and this depends on the reinforcement schedule used, to which we now turn.

The types of reinforcement for learning perceptual patterns mainly fall under one paradigm. There is a set $\mathfrak{D}$ of perceptual displays and a subset G of $\mathfrak{D}$ that we want the device to learn to select. On each trial, the device responds *yes* to the presented display d if it classifies d as a member of G, otherwise it responds *no*. If the answer is correct, it receives positive reinforcement (e_1), and if the answer is incorrect, it receives negative reinforcement (e_2). This type of reinforcement we call *standard reinforcement*.

The first goal is to find learning procedures that, given standard reinforcement, will eventually learn always to respond correctly. Second, one wants the learning procedure to be "reasonable." Although no precise definitions of "reasonable" are available, it is an important consideration, and we comment on some of the issues involved.

For perceptrons, the state of conditioning is the set of coefficients that is being used. It is convenient mathematically to order the coefficients by ordering the k partial predicates in Φ. One can then think of the coefficients and the values of the partial predicates as k-dimensional vectors. Let us choose the most general model and let the coefficients be real numbers. The states of conditioning are then vectors in the k-dimensional vector space over the real numbers, and the values of the φ's are elements of the k-dimensional vector space over $\{0,1\}$.

We let $A = \langle \alpha_1, \alpha_2, \ldots, \alpha_k \rangle$ and $\Phi(d) = \langle \varphi_1(d), \varphi_2(d), \ldots, \varphi_k(d) \rangle$ where $d \in \mathfrak{D}$ and $\mathfrak{D}$ is the domain of the φ's. This allows us to write $\Sigma_i \alpha_i \varphi_i(d)$ as $A \cdot \Phi(d)$. The response is *yes* if $A \cdot \Phi(d) > 0$ and *no* otherwise. The set of reinforcements E is $\{e_1, e_2\}$. The sample space X consists of all possible experiments; each experiment is a sequence of trials. In the present situation a trial is simply a triple $(A,d,e,)$, where A is a state of conditioning, d an object in $\mathfrak{D}$ and e a reinforcement.

A particular event is a subclass of the sample space, e.g., the event of the first reinforcement being e_2 is the set of all experiments in which e_2 occurred on the first trial. For convenience, let A_n be the state of conditioning at the beginning of trial n and d_n be the object on trial n. The dimension of the two necessary vector spaces is defined by k.

DEFINITION 13. *A structure* $\Lambda = (\mathfrak{D},\Phi,E,X)$ *is a* perceptron learning model *if and only if the following axioms are satisfied:*

(i) *If* e_1 *occurs on trial* n, *then* $A_{n+1} = A_n$.

(ii) *If* e_2 *occurs on trial* n *and* $A_n \cdot \Phi(d_n) < 0$, *then* $A_{n+1} = A_n + \Phi(d_n)$.

(iii) *If* e_2 *occurs on trial* n *and* $A_n \cdot \Phi(d_n) > 0$, *then* $A_{n+1} = A_n - \Phi(d_n)$.

We are now in a position to state the well-known learning theorem for perceptrons. (For the history of this theorem, see Minsky & Papert, 1969.)

THEOREM 14. (*Perceptron Convergence Theorem*). *For any set* $\mathfrak{D}$ *and any subset* G *of* $\mathfrak{D}$, *if there is a vector* A *such that* $A \cdot \Phi(d) > 0$ *iff* $d \in G$, *then in any perceptron learning model given standard reinforcement there will only be a finite number of trials on which the conditioning vector changes.*

The force of "any perceptron learning model" is that the choice of A_1 does not affect the result. Also, instead of saying trials on which "the conditioning vector changes" one could say equivalently "e_2 occurs" or "Ω responds incorrectly." If there are only a finite number of changes, then Ω will eventually make no mistakes. This is equivalent to saying that Ω can distinguish the elements of G *from those of* $\mathfrak{D} - G$, *provided that* every type of element in both G and $-G$ occurs infinitely often in an experiment.

The conditions of the theorem require a vector A such that $A \cdot \Phi(d) > 0$ *iff* $d \in G$. This condition can be stated in other terms by saying that G and $\mathfrak{D} - G$ are linearly separable. If the two sets are not linearly separable, the theorem simply does not apply. The vector that the procedure eventually finds will not necessarily be the vector A.

The simplicity of this learning procedure is appealing. The perceptron learns only on trials where it responds incorrectly, which is plausible. Also,

it approaches the final answer step by step, with each change bringing it closer to the final answer. Contrast this with the procedure used by a similar device, the homeostat. The homeostat uses only integer coefficients (which do not decrease its power). It learns by having an enumeration of all k-tuples of integers available, and every time it makes a mistake it goes to the next k-tuple. Sooner or later it finds one that works. It does seem implausible to have an enumeration of all k-tuples available, but this could be solved by setting up some random guessing procedure. Often, the real objection to the homeostat is that it seems to learn primarily from a "lucky guess." On the other hand, "real" learning requires a gradual acquisition of an ability. Although this intuition is widespread, it should not be accepted uncritically. The same intuition favored linear conditioning models over all-or-none conditioning models, and yet the latter have proved to fit the data of many concept-formation or concept-identification experiments better (Suppes & Ginsberg, 1963).

In any discussion of reasonableness, the question of time (measured in number of trials) is bound to arise. If a learning procedure requires an inordinately large number of trials to the last expected error, we regard it as unreasonable or impractical. Note that the perceptron convergence theorem says nothing about rate of learning. What counts as "unreasonable" is difficult to formulate precisely; it is also difficult mathematically to estimate the rate. Minor modifications (which do not affect the convergence theorem) in the choice of the starting vector or the learning procedure (e.g., add $2\Phi(d)$ rather than $\Phi(d)$) can greatly affect the learning rate. That the homeostat probably takes longer than the perceptron to learn is a definite disadvantage. We emphasize, though, that this problem is entirely different from the preceding one of "gradualness" versus "lucky guess."

We turn now to automaton learning. As mentioned in the introduction, stimulus-response learning theory is the basis of the automaton approach. After formulating an appropriate general stimulus-response model with the appropriate restrictions, the asymptotic convergence theorem is stated. However, to avoid a large number of technical details, we formulate the concepts and axioms of the theory informally. A mathematically explicit development is to be found in Suppes (1969) and Rottmayer (1970).

The following characterization of an all-or-none conditioning model is standard, except that the notion of a subtrial is introduced. A subtrial corresponds to what is usually called a trial, but no reinforcement or conditioning takes place. Conditioning occurs only after a series of subtrials, now called a trial. A subtrial corresponds to an automaton making one transition, a trial to processing a whole tape. The definition requires seven primitive concepts. There is the set S of stimuli and the set R of responses. The set E of reinforcements contains only two elements, e_1 and e_2; e_1 is the

positive reinforcer, e_2 the negative one. The fifth primitive concept is a measure μ of saliency on the set of stimuli. The concept of subtrial requires the introduction of $\mathcal{M}$ which is a sequence of positive integers m_n. Each m_n indicates the number of subtrials on trial n. This notion is necessary in defining the next primitive concept, that of the sample space X. Each element of X represents a possible experiment, i.e., an infinite sequence of trials, where each trial n has m_n subtrials. Each trial is an (m_n+2)-tuple consisting of three things: (a) the conditioning function at the beginning of the trial which is a partial function from S into R, where $C(\sigma) = r$ means σ is conditioned to r and $C(\sigma)$ undefined means σ is unconditioned; (b) m_n triples of the form (T,s,r) where T is the set of presented stimuli, s is the set of sampled stimuli and r is the response on a subtrial; and (c) the reinforcement which occurred. The final concept is the probability measure P on the appropriate algebra of events (subsets) of X. All probabilities must be defined in terms of P.

In the following axioms of the theory it is assumed that all events on which probabilities are conditioned have positive probability. There are three kinds of axioms: sampling axioms, conditioning axioms, and response axioms.

Sampling Axioms

S1. *On every subtrial a set of stimuli of positive measure is sampled with probability* 1.

S2. *If the same presentation set occurs on two different subtrials, then the probability of a given sample is independent of the subtrial number.*

S3. *Samples of equal measure that are subsets of the presentation set have an equal probability of being sampled on a given subtrial.*

S4. *The probability of a particular sample on trial n, subtrial m, given the presentation set of stimuli, is independent of any preceding subsequence of events.*

Conditioning Axioms

C1. *On every trial with probability* 1 *each stimulus element is conditioned to at most one response.*

C2. *If e_1 occurs on trial n, the probability is c that any previously unconditioned stimulus sampled on a subtrial will become conditioned to the response given on that subtrial, and this probability is independent of the particular subtrial and any preceding subsequence of events.*

C3. *If e_1 occurs on trial n, the probability is* 0 *that any previously unconditioned stimulus sampled on a subtrial will become conditioned to a response different from the one given on that subtrial, and this probability is independent of the particular subtrial and any preceding subsequence of events.*

C4. *If e_1 occurs on trial n, the conditioning of previously conditioned sampled states remains unchanged.*

C5. *If e_2 occurs on trial n, the probability is* 0 *that a previously unconditioned stimulus sampled on a subtrial will become conditioned.*

C6. *If e_2 occurs on trial n, the probability is d that any previously conditioned stimulus sampled on a subtrial will become unconditioned, and this probability is independent of the particular subtrial and any preceding subsequence of events.*

C7. *With probability* 1, *the conditioning of unsampled stimuli does not change.*

Response Axioms

R1. *If at least one sampled stimulus is conditioned to some response, then the probability of any response is the ratio of the measure of sampled stimuli conditioned to this response to the measure of all the sampled conditioned stimuli, and this probability is independent of any preceding subsequence of events.*

R2. *If no sampled stimulus is conditioned to any response, then the probability of any response r is a constant guessing probability ρ_r that is independent of n and any preceding subsequence of events.*

The conditioning method used in the theory just stated is simple. Conditioning occurs on trials where there is a correct response, and deconditioning occurs on trials where there is an incorrect response. Thus learning occurs on all trials, not only on incorrect trials as in the perceptron model. The method of learning actually resembles a homeostat more than a perceptron as far as "gradualness" is concerned.

We can prove the following theorem about stimulus-response theory as formulated above.

THEOREM 15. *If $\mathfrak{D}$ is any set of perceptual displays and G is a subset of $\mathfrak{D}$ that can be recognized by a finite automaton, then there is a stimulus-response model that can also learn to recognize G, with performance at asymptote matching that of the automaton.* (We can regard the codings of the figures in G as being a language, so that in terms of Definition 4 the

theorem says the stimulus-response model will at asymptote be weakly equivalent to any finite automaton that can recognize the figures in G.)

As far as time is concerned, the theorem is actually proved by placing a lower bound on the rate of learning. This lower bound is unreasonable, but the actual rate will be much faster. Still, the learning would probably be classified as slow, and perhaps as unreasonably slow. Minor modifications may increase the rate of learning, just as many variations are available for perceptrons.

In closing, we note that both learning procedures assumed an arbitrary starting point and simple *yes/no* reinforcement. This is the most difficult learning situation, and a slow rate of learning can probably be expected. This difficult situation is the correct one for proving asymptotic or convergence theorems. It is probably not the correct situation for the detailed study of perceptual learning. The ordinary learner does not have an arbitrary starting point, and in most situations he is given more information about what to do than simply being told *yes* or *no* after each response. Development of more complex and more realistic learning models is beyond the scope of this chapter.

References

Barlow, H. B., Narasimhan, R., & Rosenfeld, A. Visual pattern analysis in machines and animals. *Science,* 1972, **177**, 567–575.

Block, H. D. A review of "Perceptrons: An introduction to computational geometry." *Information and Control,* 1970, **17**, 501–522.

Chomsky, N. Formal properties of grammars. In R. D. Luce, R. R. Bush, & E. Galanter (Eds.), *Handbook of mathematical psychology.* Vol. 2. New York: Wiley, 1963. Pp. 125–155.

Chomsky, N., & Miller, G. A. Introduction to the formal analysis of natural languages. In R. D. Luce, R. R. Bush, & E. Galanter (Eds.), *Handbook of mathematical psychology.* Vol. 2. New York: Wiley, 1963. Pp. 269–321.

Eden, M. On the formalization of handwriting. *American Mathematical Society of Applied Mathematics Symposium,* 1961, **12**, 83–88.

Eden, M. Handwriting and pattern recognition. *IRE Transactions on Information Theory,* 1962, **IT–8**, 160–166.

Hopcroft, J. E., & Ullman, J. D. *Formal languages and their relation to automata.* Reading, Massachusetts: Addison-Wesley, 1969.

Minsky, M., & Papert, S. *Perceptrons.* Cambridge, Massachusetts: MIT Press, 1969.

Narasimhan, R. A linguistic approach to pattern recognition. Report No. 121, 1962, University of Illinois, Digital Computer Laboratory.

Narasimhan, R. Labeling schemata and syntactic description of pictures. *Information Control,* 1964, **7**, 151–179.

Narasimhan, R. Syntax-directed interpretation of classes of pictures. *Communications of the Association for Computing Machinery,* 1966, **9**, 166–173.

Needham, J. *Science and civilization in China.* Vol. IV; 2. London and New York: Cambridge Univ. Press, 1965.

Rabin, M. O., & Scott, D. Finite automata and their decision problems. *IBM Journal of Research and Development,* 1959, **3**, 114–125.

Rosenblatt, F. Two theorems of statistical separability in the perceptron. *Proceedings of a symposium on the mechanization of thought processes.* London: HM Stationery Office, 1959.

Rosenblatt, F. *Principles of neurodynamics.* New York: Spartan Books, 1962.

Rosenfeld, A. Isotonic grammars, parallel grammars and picture grammars. In B. Meltzer & D. Michie (Eds.), *Machine intelligence* 6. New York: American Elsevier, 1971. Pp. 281–294.

Rottmayer, W. A. A formal theory of perception. Technical Report No. 161, 1970, Stanford University, Institute for Mathematical Studies in the Social Sciences.

Samuel, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development,* 1959, **3**, 210–229.

Samuel, A. L. Some studies in machine learning using the game of checkers. Part II. *IBM Journal of Research and Development,* 1967, **11**, 601–618.

Shaw, A. C. A formal picture description scheme as a basis for picture processing systems. *Information and Control,* 1969, **14**, 9–52.

Suppes, P. Stimulus-response theory of finite automata. *Journal of Mathematical Psychology,* 1969, **6**, 327–355.

Suppes, P., & Ginsberg, R. A fundamental property of all-or-none models, binomial distribution of responses prior to conditioning, with application to concept formation in children. *Psychological Review,* 1963, **70**, 139–161.