

AD630125

RADC-TR-65-377, Volume I  
Final Report  
Supplement I



COMMUNICATION WITH AUTOMATA

Carl Adam Petri

SUPPLEMENT I  
January 1966

to

TECHNICAL REPORT NO. RADC-TR-65-377  
VOLUME I

Distribution of this document  
is unlimited.

CLEARINGHOUSE FOR FEDERAL SCIENTIFIC AND TECHNICAL INFORMATION			
Hardcopy	Microfiche		
\$9.60	0.75	98pp	ad
ARCHIVE COPY			

code 1  
Reconnaissance-Intelligence Data Handling Branch  
Rome Air Development Center  
Research and Technology Division  
Air Force Systems Command  
Griffiss Air Force Base, New York

COMMUNICATION WITH AUTOMATA

Carl Adam Petri

SUPPLEMENT I

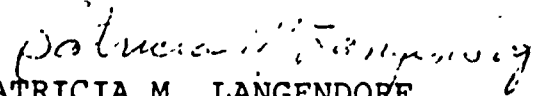
Distribution of this document  
is unlimited.

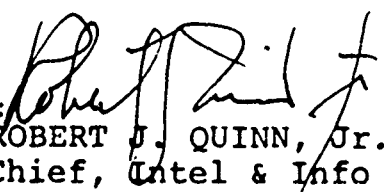
AFLC, GAFB, N.Y., 24 Mar 66-201

FOREWORD

This book forms the supplement to the final report on  $\lambda$ -Theory, developed under the Information Systems Theory Project by Applied Data Research, Inc., Princeton, N.J. (prime contractor) and Computer Associate, Inc. of Wakefield, Massachusetts (subcontractor) under Contract No. AF30(602)-3324. Miss Patricia Langendorf, Rome Air Development Center (EMIRD) was the project engineer.

This report has been reviewed and is approved.

Approved:   
PATRICIA M. LANGENDORF  
Project Engineer

Approved:   
ROBERT J. QUINN, Jr., Colonel, USAF  
Chief, Intel & Info Processing Division

FOR THE COMMANDER:

  
IRVING J. LABELMAN  
Chief, Advanced Studies Group

DECLARATION

I herewith declare myself in agreement with the translation prepared by Mr. Clifford F. Greene, Jr. of my work "Kommunikation mit Automaten" (Bonn, 1962) and grant permission for the distribution of the same, with the request that there be included with each copy of the translation a copy of the ERRATA in the form I have outlined.

sgd. Dr. Carl Adam Petri

encl: Errata (1 p.)

(Translated by Clifford F. Greene, Jr.)

## ABSTRACT

The theory of automata is shown not capable of representing the actual physical flow of information in the solution of a recursive problem. The argument proceeds as follows:

1. We assume the following postulates: a) there exists an upper bound on the speed of signals; b) there exists an upper bound on the density with which information can be stored.

2. Automata of fixed, finite size can recognize, at best, only iteratively defined classes of input sequences. (See Kleene (11) and Copi, Elgot, and Wright (8).)

3. Recursively defined classes of input sequences that cannot be defined iteratively can be recognized only by automata of unbounded size.

4. In order for an automaton to solve a (soluble) recursive problem, the possibility must be granted that it can be extended unboundedly in whatever way might be required.

5. Automata (as actual hardware) formulated in accordance with automata theory will, after a finite number of extensions, conflict with at least one of the postulates named above. Suitable conceptual structures for an exact theory of communication are then discussed, and a theory of communication proposed.

All of the really useful results of automata theory may be expressed by means of these new concepts. Moreover, the results retain their usefulness and the new procedure has definite advantages over the older ones.

The proposed representation differs from each of the presently known theories concerning information on at least one of the following essential points:

1. The existence of a metric is assumed for neither space nor time nor for other physical magnitudes.

2. Time is introduced as a strictly local relation between states.

3. The objects of the theory are discrete, and they are combined and produced only by means of strictly finite techniques.

The following conclusions drawn from the results of this work may be cited as of some practical interest:

1. The tolerance requirements for the response characteristics of computer components can be substantially weakened if the computer is suitably structured.

2. It is possible to design computers structurally in such a way that they are asynchronous, all parts operating in parallel, and can be extended arbitrarily without interrupting their computation.

3. For complicated organizational processes of any given sort the theory yields a means of representation that with equal rigor and simplicity accomplishes more than the theory of synchronous automata.

## ERRATA

- page 2, line 2: change 'be' to read 'been'
- page 3, line 10 insert 'the' between 'for' and 'con-'
- page 12, line 22: change 'signals' to read 'sequences'
- page 15, line 15: change ' $g < h$ ' to read ' $g \leq h$ '
- page 26, line 13: change 'signal, indicate' to 'signal indicating'
- page 27 line 10: change 'out-' to read 'in-'
- page 30, line 16: change 'shall' to read 'shall not'
- page 32, line 2: change 'change' to read 'charge'
- page 32, line 22: insert 'and indeterministic' after 'deterministic'
- page 37, line 13: change 'way' to read 'ways'
- page 44, line 1: insert 'of' between 'example' and 'two'
- page 47, line 12: change 'predicate' to read 'propositional'
- page 47, line 23: change 'to' to read 'from'
- page 49, line 25: change 'predicate' to read 'propositional'
- page 63, line 3: insert 'chain' after 'net'
- page 70, line 10: insert 'head' after 'reading'
- page 73, line 24: insert 'it' at end of line
- page 75, line 12: change 'and the action' to read 'and the event'
- page 78, line 4: change 'predicate' to read 'propositional'
- page 85, line 14: insert 'situation' after 'game'
- passim: change 'connection' to read 'connexion' or  
change 'connexion' to read 'connection'

(The original manuscript was submitted on July 27th, 1961. First printed in 'Schriften des Rheinisch-Westfälischen Instituts für Instrumentelle Mathematik an der Universität Bonn', Nr 2, Bonn 1962.)

## 1. Preliminaries

THIS WORK is concerned with the conceptual foundations of a theory of communication. It shall be the task of this theory to describe in a consistent and exact manner as many as possible of the phenomena that occur in the transmission and transformation of information.

Within the realm of scientific thought a key position may be ascribed to the concept of communication. It is therefore necessary to place on the means of representation and proof to be utilized in this theory the most stringent requirements of actual constructibility. However, there are two immediate applications of such a theory, namely in the design and the programming of information-processing machines. The conceptual structures of the theory should therefore not be too far removed from those currently used in the field, which have of course had their utility demonstrated.

The representation proposed here differs from the presently known theories concerning information in at least one of the following essential points:

1. The existence of a metric is assumed for neither space nor time nor for other physical magnitudes.
2. Time is introduced as a strictly local relation between states.
3. The objects of the theory are discrete, and they are combined and produced only by means of strictly finite techniques.

The basic methodological meaning of these procedures may well be of little interest to the applied scientist, but it should be noted

that this work was begun with the expressed goal of eliminating practical difficulties and that this goal has been achieved to an extent to be described in this work.

The central point of this work shall be to discover suitable conceptual structures for an exact theory of communication; somewhat less emphasis will be placed on the elaboration of this theory as a collection of theorems. However, because of the axiomatic method by which the concepts are introduced in §4 there is a ready transition to such areas as formal logic, algebra, and topology, and the theorems of these fields may find interpretation within our present theory.

The following conclusions drawn from the results of this work may be cited as of some practical interest:

1. The tolerance requirements for the response characteristics of computer components can be substantially weakened if the computer is suitably structured.
2. It is possible to design computers structurally in such a way that they are asynchronous, all parts operating in parallel, and can be extended arbitrarily without interrupting their computation.
3. For complicated organizational processes of any given sort the theory yields a means of representation that with equal rigor and simplicity accomplishes more than the theory of synchronous automata.

It should not be expected, however, that such advantages as those cited above can be had entirely without cost. This cost lies, however, not in a high degree of abstraction that would infringe on applicability,



but rather in requiring of the user a certain reevaluation of concepts as regards their simplicity. Concepts such as 'time interval', 'measurement', and 'state of a system' do not in any way constitute the building blocks of the theory. Rather, the true complexity of these concepts should be described -- 'true' in respect to those exact relationships of communication that admit of observation and control and are attendant, for example, to the process 'measurement of a time interval'.

We do not pose the question of which concepts are actually the simplest; only the successful application of the theory to be built on the proposed mathematical model will be required as justification for conceptual framework introduced herein. Nor is it claimed that it is exactly the concepts of §4 and only those that would bring us to the goal, nor is it asserted that the content of this work represents a formally closed theory admitting of no further discussion. In the framework of what is known our mode of procedure might be described as follows: the propagation of physical effects is investigated from the standpoint of combinatorial topology and is interpreted as the switching logic of totally asynchronous automata.

## 2. A Critique of the Theory of Automata

The theory of automata has produced a number of different abstract models of information-processing machines. These various models have the following features in common: 1) one proceeds from the fiction that it is at all times meaningful to speak of the total state  $Z$  of a system  $S$  at a time  $t$ . 2) A process in  $S$  can then be described by means of a function  $f$  such that  $Z = f(t)$ . 3) The domain over which  $f$  is defined is the set of real numbers or a subset thereof. Classical physics, too, takes the above

outlook, and it was only later, as a result of the theory of relativity, that the utility of this approach was questioned.

Thus, to drop the above-described fiction and to use something else as the basis of our approach is to depart considerably from the established mode of thought. It therefore seems necessary a) to show the limitations that this fiction imposes on the abstract model; b) to show that all of the really useful results of automata theory may be expressed quite effortlessly by means of new concepts to be introduced herein and that the results therewith retain their usefulness; and finally c) to show that the new procedure has definite advantages over the older ones.

We shall begin by illustrating the difficulties that have arisen when the abstract models of automata theory are confronted with reality.

We shall argue the following points:

1. We assume the following postulates: a) there exists an upper bound on the speed of signals; b) there exists an upper bound on the density with which information can be stored.
2. Automata of fixed, finite size can recognize, at best, only iteratively defined classes of input sequences. (See Kleene [11] and Copi, Elgot, and Wright [8].)
3. Recursively defined classes of input sequences that cannot be defined iteratively can be recognized only by automata of unbounded size.
4. In order for an automaton to solve a (soluble) recursive problem, the possibility must be

granted that it can be extended unboundedly in whatever way might be required.

5. Automata (as actual hardware) formulated in accordance with automata theory will, after a finite number of extensions, conflict with at least one of the postulates named above.

In short: the theory of automata is not capable of representing the actual physical flow of information in the solution of a recursive problem.

In order to clarify the goal of our considerations, let us examine our arguments in terms of the practices attendant to computer construction, before we actually proceed with the execution of these arguments.

The logical plan of a computer specifies a finite number of states that the computer must assume at particular points in time or within particular time intervals. To these logical states there must correspond physically distinguishable states of the computer; these machine states may be regarded as belonging to the cells of a phase space. We shall now attempt to solve a recursive problem on such a machine. We can predict nothing regarding the storage that the machine will require for the solution of the problem. We either reach a solution or come to a stage in the solving at which the machine signals of a further storage requirement. Assuming that we do not wish to halt the process and forego all the benefits of the uncompleted attempt at solution, there are two possible courses of action.

The first alternative would be to bring about an output of the information onto some external storage medium, with the possibility of re-

loading the machine. We then manipulate this storage medium in such a way that, relative to the machine, it will constitute an external memory. However, in that case the machine itself solves only a trivial, at best iterative portion of the recursive problem.

The second alternative would be to construct a second machine, one that would either substitute for our manipulation of the data (by increased storage) or would carry out this manipulation for us. If this machine is in turn described by a logical plan, then this extension may be viewed as the addition of finitely many cells to the phase space. There now arises the problem of coordinating the action of the two machines. In general, however, this coordination cannot be effected without a modification or rebuilding of the original machine, for if we assume that the temporal separation of the states is defined by an oscillator with a certain base frequency, then the signal transmission times possible after the extension of the machine may exceed the basic period of the oscillator; in any case, the basic period will be exceeded after some finite number of extensions. We must therefore lower the base frequency, and after some finite number of frequency reductions the metric properties of the components will have to be changed to fit the new base frequency. (For those sorts of coupling for which no upper bound on the frequency is prescribed, the modifications of the components are required in order that the number of misfunctions per cycle be held within certain bounds.)

An analogue to these conditions is provided by the addressing of a computer memory having fixed word length: with regard to addressing, a computer memory is not unboundedly extensible, for after a certain finite

number of extensions we come to a point at which the information required to designate a memory cell cannot be contained within one computer word. We must then use other means of attacking the problem or increase the word length and, thereby, fundamentally alter the structure of the memory.

We shall treat these difficulties as a problem of the logical structure of communication, with the aim of generalizing the form of communication in such a way that these difficulties will disappear.

We shall now pursue the individual points of the argumentation sketched above.

Point 1. We shall forego any attempt to use simpler assumptions to derive the existence of an upper bound on signal transmission speed or on information density; we shall therefore introduce postulates regarding these bounds. It must be admitted that this argumentation becomes untenable if a means is found by which signal transmission speed and information density might be increased without limit. By way of an explanation, then, let it be added that we regard the speed of light as an upper bound for signal transmission speed. The second postulate serves to express the impossibility of constructing a physical object of finite volume that could be used as a memory for arbitrarily large amounts of information. This postulate gains some plausibility, perhaps, if we associate it with the quantum postulate.

If we designate the stored information by means of the impulse coordinates of a specified set of material particles, for example, then the amount of information to be stored will prescribe a certain exactness for the measurement of the impulse coordinates. As the amount of

information increases the impulse coordinates must be specified more and more exactly, but according to the quantum principle the inexactness of the positional coordinates will increase beyond all bounds. Therefore, the spatial domain in which the information is to be protected from disturbance, i. e. the volume of the memory, will have to be increased beyond all bounds.

Point 2. Initially we shall speak only of deterministic, synchronous, digital automata. For such automata one proceeds from the assumption that they enter well-defined states at specific times  $t_i$ , where  $i \in \{\dots -2, -1, 0, 1, 2, \dots\}$ ; the state at time  $t_{i+1}$  and the output signal at time  $t_i$  are single-valued functions of the state and the input signal at time  $t_i$ . The domain of values of the input and output signals is assumed to be finite. If the set of possible states that the automaton can assume is specified and finite, then we speak of a finite automaton.

Automata may now be characterized in several ways (v. Lee [11]):

- a) by specifying the connection between the states and the signals at times  $t_i$  and  $t_{i+1}$  for all  $i$ ;
- b) by writing a program to simulate the automaton;
- c) by specifying the structure of the composition from idealized components of the automaton;
- d) by specifying a mapping of the input-signal sequences onto the output-signal sequences.

For our purposes, the last-named method of characterization, especially, is essential. We may ask ourselves what properties of an input sequence can be recognized by an automaton, wherein a property is said to be recognizable if it can be uniquely brought into correspondence with some property of individual output sequences in a manner to be shown below.

Let  $X = \{a, b, c, \dots\}$  be the finite set of input signals, and let  $Y = \{\alpha, \beta, \gamma, \dots\}$  be the finite set of output signals; let  $\mathcal{U}Y$  be the set of all subsets of  $Y$  (including  $\emptyset$  and  $Y$ ). The subsets of  $Y$ , that is, the elements of  $\mathcal{U}Y$ , characterize all possible properties of individual output signals. For example, the subset  $\{\alpha, \beta\}$  designates the property (of a signal) of being either  $\alpha$  or  $\beta$ . These properties form a boolean ring.

We now wish to characterize the properties of input sequences. In line with our task of investigating communication relationships, we shall view the input sequences as messages to the automaton. Since  $\mathcal{U}Y$  contains only finitely many elements, the automaton can recognize at most the same number of properties.

The set of messages to the automaton forms, with respect to concatenation, a free semigroup with identity with the set  $X$  of generators, which set we wish to view as the alphabet of a formal language. Since we can consider only the formal properties of messages, we define the expressions in this language purely by means of syntax. For the representation of the syntax, we make use of a formal metalanguage the alphabet of which contains all the elements of  $X$ , that is  $a, b, c, \dots$ , and, in addition to these, the symbols  $\vee, /, \ast$ , and  $\emptyset$ .

An expression in the metalanguage designates a set of input sequences. This metalanguage is chosen in such a way that its own expressions belong to a very simple syntax, namely that of a simple phrase structure grammar (see Chomsky [7]), or, more generally, so that the expressions are the formulas of a very simple combinatorial system (see Post [19]).

In the definition of a combinatorial system we follow Davis [10] (our paraphrasis or undesignated quotation being derived from material on pages 83ff. of [10]): A word is a finite, possibly empty, sequence of

symbols. A *production* is a two-place predicate between words and is designated by an ordered sextuple of words  $g, h, k; \bar{g}, \bar{h}, \bar{k}$ . For a given ordered pair of words  $F$  and  $G$ , the predicate is true if and only if there exists words  $P$  and  $Q$  such that

$$F = g P h Q k \quad \text{and} \quad G = \bar{g} P \bar{h} Q \bar{k}.$$

We shall then say that  $G$  is a *consequence of  $F$  with respect to the production*. A *combinatorial system*  $\Gamma$  consists of a single non-empty word called the *axiom of  $\Gamma$*  and a finite set of productions called the *productions of  $\Gamma$* . The alphabet of  $\Gamma$  consists of all symbols that occur either in the axiom of  $\Gamma$  or in the  $g, h, k, \bar{g}, \bar{h}, \bar{k}$  that define the productions of  $\Gamma$ . A *proof (or derivation) in a combinatorial system  $\Gamma$*  is a finite sequence of words  $F_1, F_2, \dots, F_m$  such that  $F_1$  is the axiom of  $\Gamma$  and such that for all  $i$  ( $1 \leq i \leq m$ ),  $F_i$  is a consequent of  $F_{i-1}$  with respect to the productions of  $\Gamma$ . A word is a *theorem in  $\Gamma$*  if it is the final step of a proof in  $\Gamma$ . A *formula in  $\Gamma$*  is a theorem that has no consequent with respect to the productions of  $\Gamma$ .

To return now to the discussion of our metalanguage -- our expressions for the description of input sequences are to be formulas in the following combinatorial system CS1:

Axiom:  $\Sigma$

Productions:  $g, k, \bar{g}, \bar{k}$  are empty  
 $h = \Sigma$ .

The productions differ then only in  $\bar{h}$ , and finding a consequent consists in carrying out the substitution  $\Sigma \rightarrow \bar{h}$ . Accordingly, we shall describe our productions as follows:

$\Sigma \rightarrow \nabla \Sigma \Sigma$

$\Sigma \rightarrow / \Sigma \Sigma$



$$\left. \begin{array}{l} \Sigma \rightarrow * \Sigma \\ \Sigma \rightarrow \emptyset \\ \Sigma \rightarrow a \\ \Sigma \rightarrow b \\ \vdots \end{array} \right\} \text{enumeration of the elements of } X.$$

To each formula of this system we shall assign a set of sequences of  $X$ -elements (i. e. of input signals) according to the following rules:

- $\emptyset$  designates the empty set (signal),
- $a$  designates the set containing the sequence 'a' of length 1,
- $b$  correspondingly, ...,
- $\vee \Sigma \Sigma$  designates the union of two sets each derived from a  $\Sigma$ -symbol by substitution,
- $/ \Sigma \Sigma$  designates the set of those sequences of length  $\ell$ , where  $\ell = \ell_1 + \ell_2$  ( $\ell_1 > 0, \ell_2 > 0$ ) such that the first  $\ell_1$  elements of the sequence constitute a sequence belonging to the set assigned to the (partial) formula derived from the first  $\Sigma$ -symbol by substitution, and similarly, such that the last  $\ell_2$  elements of the sequence constitute a sequence belonging to the set assigned to the (partial) formula derived by substitution from the second  $\Sigma$ -symbol,
- $* \Sigma$  designates the set of exactly those sequences for which there exists a positive integer  $n \neq 0$ , such that the sequence can be partitioned into  $n$  partial sequences each of which belongs to the set assigned to the partial formula derived by substitution from the  $\Sigma$ -symbol.

In accordance with the above we designate  $\vee$  as the *selection operator*,  $/$  as the *sequence operator*, and  $*$  as the *iteration operator*.

We now give two theorems, T1 and its converse, T2.

**Theorem T2:** For every finite set of finite formulas of the combinatorial system CS1 there may be specified a finite, digital, de-

terministic, synchronous automaton such that to each formula of the set there corresponds a property of the input sequences recognizable by the automaton.

Theorem T1: All properties of the input sequences recognizable by a given finite, digital, deterministic, synchronous automaton are specified by a finite set of finite formulas of the combinatorial system CS1.

These theorems derive their significance primarily from the definition of *recognizability*, which we shall now state and discuss.

Def: A property  $P$  of an input sequence is *recognized* by an automaton if there exists a property  $Q$  of individual output signals such that the output signal has the property  $Q$  at time  $t$  if and only if the input sequence whose last symbol appears at time  $t$  has the property  $P$ .

The usefulness of this definition should be elucidated briefly. We must regard the properties of individual input signals as distinguishable in all cases, and therefore, colloquially speaking, as recognizable. This is inherent in the nature of digital signals. Beyond this we are interested in the properties of sequences of such input signals, for example the property of containing only the signals  $a$  and/or  $b$ , which would be designated by  $\forall ab$ . Such properties could be mapped onto properties of output signals (see [3], [8], [16], and [15]); in contrast to this, however, we are concerned with exactly those properties that are expressed in individual output signals (generalized after [11] and [8]). In the case of deterministic automata, it is obviously possible to reconstruct the properties of the output sequences from the properties of the input sequences, but such reconstruction is of no interest here since it does

not contribute to the characterization of the automaton.

We have regarded the set of possible output signals as specified in advance. One might ask, however, whether there perhaps exist internal signals not belonging to the set of output signals but yet capable of distinguishing even those input properties that are not distinguished by the output signals. It would then be questionable whether such properties should be regarded as recognized (or recognizable). We free ourselves of this dilemma by stipulating that the output signal at time  $t$  be the ordered  $p$ -tuple of all  $p$  observable elementary signals in the automaton at time  $t$ . In any particular instance, then, it must be established in advance which elementary signals are to be considered observable. With this stipulation the theorems become applicable both to automata conceived of as black boxes as well as to automata with a physical-structural type of characterization.

For a proof of theorem T2 we may refer to the work of Copi, Elgot, and Wright [8], in which there is given an algorithm for the synthesis of logical nets that fulfill the requirements of the theorem. (Following a usage that is of some advantage if the formulas serve only an explanatory function, those authors use the following productions in a completely equivalent manner:

$$\begin{array}{lll} \Sigma \rightarrow (\Sigma \vee \Sigma) & \text{instead of} & \Sigma \rightarrow \nabla \Sigma \Sigma \\ \Sigma \rightarrow (\Sigma \cdot \Sigma) & " & \Sigma \rightarrow / \Sigma \Sigma \\ \Sigma \rightarrow \Sigma^* & " & \Sigma \rightarrow \# \Sigma \\ \Sigma \rightarrow \Lambda & " & \Sigma \rightarrow \emptyset \quad .) \end{array}$$

The authors utilize elements for conjunction, disjunction, negation, and time delay in the 'construction' of logical nets. As properties of the output signals only the values of the elementary signals are required.

From this characterization of an automaton as a logical net, one can with no further difficulty make the transition to other representations, for example to that of state graphs.

Copi, Elgot, and Wright likewise supply the proof of theorem T1 in all its essentials (see [8] p. 187) and only a couple supplementary remarks are required to justify the theorem in the form used here.

There is no loss of generality in assuming the range of values of the elementary signals to be the set  $\{0, 1\}$ . For deciding whether a given signal has a particular property, a signal of this range is sufficient; conversely it is well known that a signal of the greater range  $\{a_1, a_2, a_3, \dots, a_n\}$  can be represented by a  $k$ -tuple of binary elementary signals, where  $2^k \geq n$ . We can suppose that  $k = n$  and that the binary encoding of the input signals is continued throughout the entire automaton. Further, there is no loss of generality caused by the authors' treatment of only those automata that are characterized by the way(s) in which their elementary switching elements are combined. If, instead, the finite state graph of the automaton is given, then there can always be constructed a switching circuit equivalent to the state graph. Even beyond this, McNaughton and Yamada state algorithms permitting the formulas (regular expressions) of the output signals to be found directly from the state graphs (and conversely)(see [14]).

Finally, we speak more generally of properties of an output signal, whereas Copi, Elgot, and Wright treat only the binary properties (i. e. the values at time  $t$ ) of the individual elementary signals at the output. The theorem has, of course, nothing to do with special coding, and we put forth our generalization in order that it be independent of the encoding of signals in its wording as well. Consequently, our theorem T1

becomes somewhat stronger, and our theorem T2 somewhat weaker, than the corresponding theorems of Copi, Elgot and Wright. Our theorem T1 therefore requires an additional step of proof, whereas for T2 we need only note that, by means of the above-mentioned synthesis algorithm, the properties to be recognized are in fact already expressed by means of the binary elementary signals. The required additional step of proof reads:

To prove: Every subset of the set of output signals can be represented by a CS-1 formula if every output signal is represented by such a formula:

Proof: As was mentioned above, the subsets form a boolean ring of  $2^k$  elements with the  $k$  generators  $\alpha, \beta, \dots$ . Let  $\emptyset$  be the formula representing the empty subset, and let  $F_\alpha, F_\beta, \dots$  be the representations of  $\alpha, \beta, \dots$ . We assume the assertion to be true for all subsets of  $g < h$  elements; now if  $H$  is a subset with  $h$  elements and  $F_H$  its representation, then the representation  $F_J$  of a subset  $J$  of  $h+1$  elements, such that  $J \supset H$ , is given by the formula  $F_J = \forall F_H F_V$ , where  $F_V$  is the additional element of  $J$  not in  $H$ . However, this is again a formula of the combinatorial system, and since every subset is finite, containing at most  $k$  elements, the formula representing any given subset is also finite.

We may now say that an automaton of the type considered can receive arbitrary input sequences containing messages in some formal language and that it can evaluate such messages, that is to say, can produce an effect

well-defined with respect to manner and time, however, only insofar as the syntax of the formal language is expressible by means of formulas of the system CS1.

Such a syntax as is expressible by a finite set of CS1 formulas we shall refer to as an *iterative syntax*, for it is primarily  $*$ , the iteration operator, that defines the power of such a system. For if an automaton has but one state and is therefore equivalent to a closed circuit, then it can recognize only individual input signals: the syntax of any of the possible input languages is 'contained' in CS1, and only the  $\forall$  operator is required. If the automaton contains delay elements then also the operator  $/$  is required, and, finally, to express the feedback of signals, the operator  $\times$  is required.

For the sake of brevity we shall also speak of *iterative messages* in *iterative languages*, i. e. messages in languages expressed in an iterative syntax (as defined above). Further, by an *iterative problem* we shall mean a problem the treatment of which consists in considering as the input sequence to an automaton some arbitrary symbol sequence (the *problem statement*) recognized according to an iterative syntax; the *solution* to such a problem will comprise the sequence of recognitions, i. e. the output sequence.

A syntax that cannot be specified (expressed) by a finite number of CS1 formulas we shall call *irregular*. Irregular syntax is encountered, for example, in any language in which brackets (in their usual meaning) are admitted without limit, as in most mathematical formula languages, for example. Further, the syntax of ALGOL as well as of natural languages is irregular. The interest in irregular languages in connexion with automata derives primarily from the necessity of solving recursive problems

(insofar as such problems can be solved with the aid of a Turing machine). It might be mentioned, however, that the concept of *conflict* to be introduced below (Part 4) allows for the treatment of a yet more general class of irregular problems.

It is felt, then, that the second point of our argument has been elucidated sufficiently: Automata of fixed, finite size can recognize, at best, only iteratively defined classes of input sequences.

Point 3. We shall now prove our third assertion, namely that recursively defined classes of input sequences that cannot be defined iteratively can be recognized only by automata of unbounded size, by showing that with even a slight extension of the combinatorial system that served to characterize the languages for finite automata, an automaton of fixed, finite size will not suffice for deciding whether a given message is a member of a given set, although a Turing machine (with its tape) would be adequate. We shall imagine this tape as consisting initially of but one square, with the condition that any displacement (of the tape) that would result in there being no existent square under the reading head will cause a square (always with the same initial symbol) to be produced under the reading head. The size of this machine together with its tape is then unbounded.

We wish, first of all, to elucidate the expression *recursive message*; by this term is meant, then, a message in a language with a *recursive syntax*, that is, in a *recursive language*. We shall call a syntax *recursive* if it contains a *recursion operator*, which will be defined directly. Such a syntax may be generated by the combinatorial system CS2 as follows:

Axiom:  $\Sigma$

Productions:  $\Sigma \rightarrow \forall \Sigma \Sigma$

$\Sigma \rightarrow / \Sigma \Sigma$

$\Sigma \rightarrow R$

$\Sigma \rightarrow a$

$\Sigma \rightarrow b$

$\vdots$

} enumeration of the elements of  $X$

We shall now give an interpretation to the formulas in CS2. To each formula of the system there is assigned a set of sequences of  $X$ -elements in the following manner:

$a$	}	exactly as in CS1
$b$		
$\vdots$		
$\forall \Sigma \Sigma$		
$/ \Sigma \Sigma$		

We shall give a constructive interpretation for the  $R$  operator by considering a Turing machine with the possibility of input and output; this machine is to be capable of producing on its tape every element of the set of signal sequences; this it does by working through the formula specified on its tape initially and, in the event of choice, by requiring of its environment (that is, of some external source) the information necessary for reference to a definite element. This input will be in the form of a sequence of 0's and 1's. (The same sort of procedure could be applied in a simpler manner to the formulas of CS1.) By means of this procedure we also obtain a mapping between the sequences of 0's and 1's, which correspond to the successive choice points associated with a given formula, and the sentences of a recursive language. It should be noted that the significance of this mapping lies deeper than the mere possibility of encoding the individual symbols of the languages with 0's and 1's.



The machine reads the formula from left to right and proceeds as follows: initially the tape is to contain the formula and, behind the formula, the bracket pair [] (for the reception of the generated sequence). The machine is now ready to interpret the first symbol of the formula.

Symbol Interpreted	Operation
/	None (interpret next symbol to the right)
a } b } ⋮ }	The symbol under consideration replaces ] and ] is placed behind it; the symbol token interpreted remains unchanged.
∇	The machine interrogates the environment as to whether a recursion step is to be executed. (Reply 0 = no, 1 = yes) If a <i>no</i> reply is received, the machine interprets the next partial formula and skips over the partial formula following that one. On receipt of a <i>yes</i> reply from the environment, the machine skips over the next partial formula and treats the partial formula following that one.
R	This token is replaced by the shortest ∇-partial formula in whose second sub-partial formula it (this R) appears. The interpreting is then continued with the first symbol of the sub-partial formula that has been inserted.

The attempt to interpret the bracket [ either ends the interpreting or enables the contents of the [] to be released as output. If, in the interpretation of R, no shortest ∇-partial formula is found, or if some particular recursion step is required indefinitely, then the interpreta-

tion procedure comes to no conclusion and no output is produced. The interpretation of  $R$  is thus a generalization leading to the same effect as the interpretation of  $\emptyset$  in CS1.

Furthermore, all sets of sequences representable in CS1 can be represented in CS2, for the set produced by  $\forall \Sigma \Sigma$  in CS1 is of course exactly the set produced by  $\forall \Sigma \Sigma$  in CS2, if no  $R$  or  $\emptyset$  is encountered in the evaluation of the expression. Further, the CS1 formula  $\ast \Sigma$  can be expressed in CS2 by means of the formula  $\forall \Sigma / \Sigma R$ .

We shall not consider the programming details of the interpretation process; we shall mention only that additionally a bounded set of symbol forms or tapes should be available, with which it is possible to mark the location of observed symbols and of partial formulas to be skipped or recognized. The recognition of partial formulas can be effected in a simple manner by counting: let a counter be set to '1' and let a mark be placed before the string to be tested. For  $\forall$  or  $/$  let the counter add '+1' to its contents, for  $R, a, b, \dots$  let '-1' be added. As soon as the counter shows '0' let an additional mark be placed behind the observed symbol. By this means, the (unique) partial formula beginning at (that is, just after) the first mark is delimited. For each symbol of a formula there begins a partial formula with that symbol as its beginning. These delimiting marks are not copied during the insertion of a formula.

It is now easily recognizable that there exist recursive languages that are irregular. This may be elucidated by a simple example; let us consider a sequence set containing the following elements:

0  
101  
11011  
1110111  
111101111      et cetera.

If we were able to specify exactly this set by means of a (finite) CS1 formula, then we could also specify a finite switching network that would recognize whether a sequence had the property of belonging to this set and would therefore produce an output signal at time  $t$  with some particular property if and only if the input sequence upto time  $t$  belonged to the set. We now consider the switching network at that point of time at which the first 0 appears at the input; at this point of time it is established which element of the set can stand at the beginning of the input sequence. The subsequent activity of the switching network must be different for each element of the set, and, therefore, at the appearance of the first 0, the network must be in a different state for each element of the set. However, since the network is capable of only a certain finite number of states, this is impossible.

On the other hand, this set *can* be produced by the CS2 formula

$$F1: \quad \forall 0 // 1 R 1$$

and it is clear that there exists a Turing machine that recognizes this set (see Davis [10]).

If we examine the following formula from CS1,

$$// * 1 0 * 1$$

then it can be seen that although this formula does indeed produce a proper 'superset' of the set of sequences generated by  $\forall 0 // 1 R 1$ , it nevertheless does not generate exactly *the* set; in combinatorial system CS1 there is no way in which it is possible to express that the number of iteration steps must be the same for each of the two \* in the formula (see Curry and Feys [9]). Therefore, in the theory of finite automata it is *not even possible to express the equality of two natural numbers.*

Point Four. In order for an automaton to solve a (soluble) recursive problem, the possibility must be granted that it can be extended unboundedly in whatever way might be required.

These matters probably appear trivial to most persons of a practical outlook. Since there is, after all, no possibility of constructing an 'arbitrarily large' or even 'infinite' automaton, one simply accepts things as they are. A finite physical structure can 'store information' only in finite amounts; it is capable of only finitely many distinguishable states. From this it follows that, as a matter of principle, the automaton cannot recognize irregular messages, if it is to be regarded as a synchronous structure. It is only this last assumption, usually unpronounced, that we wish to refute.

As regards the performance capabilities of an automaton, it is by no means immaterial at what place we bring the existence of bounds into play. This is apparent whenever the structure of an automaton or of a program is itself the object of a data process, a procedure that, similarly, is to be executed by an automaton. This step is not generally practicable.

As an example, let us examine the translation of ALGOL text into machine coding. ALGOL is an irregular language -- to see this one need only notice the inductive declarations for 'arithmetic expression' or for 'block' etc. To construct an ALGOL translator is to produce a program for the translation of a substitute language (iterative pseudo-ALGOL syntax). The user of the translator must then know the substitute language if he is to anticipate the activity of the translator. This requirement does not seem to be serious; the user believes he understands the necessity of limitations, such as the following: 'no more than 500 bracket pairs

may be nested' or 'no more than 10000 names may appear in a program' or implicitly 'no more than 50000 instructions may appear in a program'.

The difficulty, however, is the following: as the complexity of the use of the machine increases, there arises between the machine and the user an insoluble problem of agreement, for the user *necessarily* requires a recursive description of the language he uses for expressing his problem or in which another user has formulated a problem, if he wishes to 'understand' the problem. In the simplest case he must know that within a domain a name token refers to the same object as a second token of the same form (and actually more generally, for arbitrarily many different names). In an automaton there can be no discussion of such 'comprehension', as was shown by the example  $\forall 0 // 1 R 1$ . It is therefore *impossible* for the machine and the user to assume the same complete description of a language as a common basis of communication.

Thus it is not possible to give a complete specification of a language such as ALGOL, its semantics included (with the aid of equivalence relations) without leaving the realm of automata theory. (Success *can* be had, however, with a finite number of CS2 steps.)

If a generalization of the conceptualizations of automata theory can be achieved such that the theory a) is at least as well suited to the physical postulates as the previous models and b) permits communication in CS2 languages, then therewith the situation of practical data processing will be fundamentally changed. The decisive point here is the fact that the set of CS2 formulas (and therewith the 'set of recursive languages') can be produced from a single CS2 formula in the same way as the set of symbol sequences of a language can be produced from the formulas of combinatorial system CS2.

This is not the case for CS1, for, viewed as symbol sequences, the set of CS1 formulas, just as the set of CS2 formulas, does not belong to any regular metalanguage. In contrast, it is easily shown that there exist finite specifications for symbol-sequence interpretation which allow an isomorphism between CS2 and the syntax of the descriptive language (meta-CS2).

In the interpreted combinatorial system CS2 we therefore have the possibility of describing the syntax of the metalanguage with the same means of expression as is used for the language itself. If we use the 'encapsulation' of CS2 symbol forms to designate the isomorphic assignment of CS2 symbol forms to meta-CS2 symbol forms, then we shall have the following as an alphabet:

$$\{ \textcircled{0}, \textcircled{1}, \textcircled{\vee}, \textcircled{/}, \textcircled{R} \}.$$

Every CS2 formula on the alphabet  $\{ \textcircled{0}, \textcircled{1} \}$  is then derivable in finitely many steps from the following formula:

$$F2: \quad \vee \vee \vee \textcircled{0} \textcircled{1} \textcircled{R} / / \vee \textcircled{\vee} \textcircled{/} R R .$$

Proof: We shall number the individual symbols of the formula and explain them as follows:

$$\begin{array}{cccccccc} \vee & \vee & \vee & \textcircled{0} & \textcircled{1} & \textcircled{R} & / & / & \vee & \textcircled{\vee} & \textcircled{/} & R & R \\ 1 & 2 & 3 & & & & 1 & 2 & 4 & & & 1 & 2 \end{array}$$

$\vee_3$  specifies the choice of an alphabet symbol;

$\vee_2$  specifies the choice between  $\textcircled{0}$  on the one hand and an alphabet symbol on the other;

$\vee_4$  specifies the selection of an operator  $\textcircled{\vee}$  or  $\textcircled{/}$ ;

$/_2$  specifies that a formula is appended to an operator and that this formula should form the leading partial formula of that operator;

$/_1$  specifies that a formula is attached to the first partial formula of an operator;

$R_1$  and  $R_2$  depend only on  $\forall_1$  and therefore designate copies (of the above formula) that are to be interpreted;

$\forall_1$  specifies the recurrent application of the CS2 productions  $\Sigma \rightarrow / \Sigma \Sigma$  or  $\Sigma \rightarrow \forall \Sigma \Sigma$ .

Now the symbol forms  $\textcircled{0}$ ,  $\textcircled{1}$ ,  $\textcircled{\forall}$ ,  $\textcircled{/}$ ,  $\textcircled{R}$  must yet be expressed by suitably chosen sequences of the two as yet unused symbols of the metalanguage, 0 and 1. Here we have some freedom. We shall choose an assignment the effects of which are easily surveyed, and we shall suppose that the following substitutions are made in the formula F2:

- $\textcircled{0} \rightarrow 0$
- $\textcircled{1} \rightarrow / 1 0$
- $\textcircled{\forall} \rightarrow // 1 1 0$
- $\textcircled{/} \rightarrow /// 1 1 1 0$
- $\textcircled{R} \rightarrow //// 1 1 1 1 \forall 0 / 1 R$

There then exists a finite switching network with the two inputs {0, 1} and the six outputs  $\{t, \textcircled{0}, \textcircled{1}, \textcircled{\forall}, \textcircled{/}, \textcircled{R}\}$  that assigns to each sequence of the metasymbols {0, 1} a sequence of object symbols (defective through  $t$ ) and therewith a sequence of CS2 formulas; this network is to be seen in Figure 1. The end of the first formula of the sequence can, if it exists, be recognized by the counting technique described above.

Then the description of the modus operandi of the machine that is to interpret the formula is merely a more exact constructive formulation of what we have understood by an interpreted combinatorial system with correspondingly simple productions. A machine for interpreting CS2 expressions we shall call an *R-machine*. This machine first receives a CS2 formula; it then receives an additional sequence on the alphabet {0, 1} for controlling the course of the recursions -- we can suppose that this

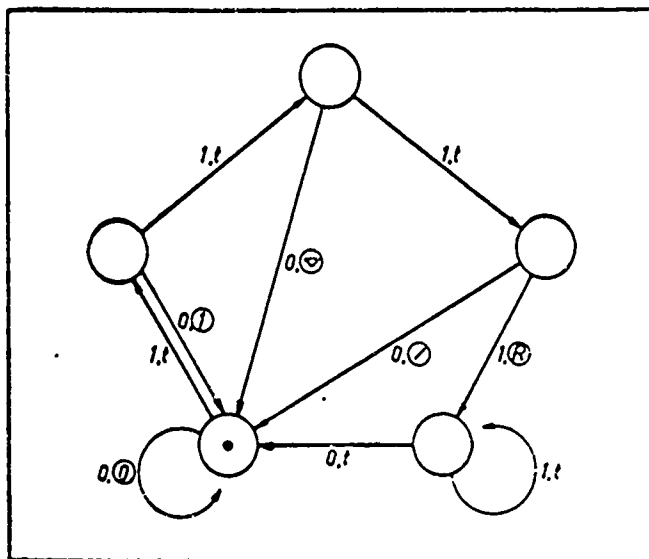


Figure 1.

sequence of 0's and 1's stems from another R-machine. Applying a formula to a sequence we shall call an *application*, as in combinatory logic (see Curry [9]). The transition from the metalinguistic expression to the linguistic expression is then reduced to this two-place operation. Since the R-machines differ only in their CS2 formulas, it becomes possible to describe them solely by means of such formulas, and even machines of unbounded size may be described by finite formulas.

In this description all structural differences between messages and automata disappear. We could supplement the programming of the R-machines in such a way that the machine would not merely come to a stop after releasing its output, but rather would erase its tape (which erasure can be effected in a finite number of steps since otherwise no output could have taken place) and, by giving some signal, indicate that it is prepared to receive a new CS2 formula. We could then follow the flow of information in an applicative system (see Curry, [9]): this system would consist of a set of objects (namely finite formulas) on which there is a three-place relation: Formula 1 applied to Formula 2 yields Formula 3.



Since, as was mentioned above, the interpretation of CS2 embraces with it that of CS1, an  $\mathcal{R}$ -machine presented with a token of F2 can produce the description of any finite automaton (switching circuit), and, further, it can produce any finite sequence of 0's and 1's (through the appending of a 0). In particular, the description of a universal Turing machine can be generated, and therewith, therefore, every Turing machine (with suitable arrangements regarding coding).

Now one knows from the theory of Turing machines (see Davis [10]) that there can exist no algorithm (itself to be represented by a Turing machine) that, presented with Formula 1 ('Program') and Formula 2 ('output data'), allows a decision as to whether Formula 3 (application result) exists, i. e. whether the sequence representing it (F3) is finite; see Davis [10] p. 70. For the attempt to execute an application there is in general no guarantee of success and actually no guarantee that success can be achieved in any previously specified number of computational steps. If one is interested in the result of an application then the only thing to do is to start the process and then to wait for a completion signal, about which we can in general say nothing regarding its appearance within any given time span. This well-known fundamental fact of irregular communication, along with physical considerations, should serve as further impetus for choosing the primitive 'switching elements' in such a way that only that property is required of them that is indispensable only in the total system and can be effectively preserved even in an unbounded system: namely the invariance of certain combinatorial-topological connexions between neighboring elements.

Point 5. Automata formulated in accordance with automata theory will, after a finite number of extensions, conflict with at least one of

the postulates named above. John von Neumann (in unpublished work, according to a note in [2]) and A. W. Burks [2] have specified the construction, from logical elements ( $\wedge$ ,  $\vee$ ,  $\ddagger$ ) and delay elements, of a memory that has the same modus operandi as the tape of a Turing machine. In that work, infinitely fast signals are presupposed, whereby the memory contradicts one of the postulates from the very start. The construction can now be modified by adding a rule (namely one corresponding to the postulate of finite signal velocity) to the rule for element connexion, the latter rule being defined over a planar square net; this is done in such a way that any signal connecting the point with (integral) net coordinates  $(x_1, y_1)$  with a point  $(x_2, y_2)$  will have to pass through at least  $D$  delay elements, where

$$D = d \cdot (|x_2 - x_1| + |y_2 - y_1|), \text{ where } d \text{ is constant } 0 < d < 1.$$

Then, in the case of growing memories, the number of delay elements is not bounded, and, therewith, their distance from each other is not bounded either. If the distance were bounded then we could achieve synchrony of the delay elements by means of time signals, so that the inexactness of the delay time  $\Delta\Delta t$  could in no case lead to the situation that a later object signal would overtake an earlier one on another path.

These time signals would now have to connect arbitrarily distant points, and therefore they must themselves be subject to the supplementary connexion rule (regarding signal speed). And therewith new delay elements have to be added, the number of which is again proportional to the observed distance, and these new delay elements will themselves have to be synchronized with time signals and laid upon free lattice points of the net.

Even if we grant the possibility of only linear extension

of the automaton that is to be extended and even if we admit an  $n$ -dimensional net, then for every  $k$  and  $d$  there can be found an  $\ell$  such that

$$N > k \cdot L^n \quad \text{if } L > \ell \quad (k > 0, \quad 0 < d < 1)$$

where  $L$  is the length of the linear arrangement of switching elements that is to be constructed and is to be established by the enumeration of  $(n-1)$ -dimensional planes, and where  $N$  is the total number of delay elements required by the connexion rules. Thus the elements cannot be placed in the net even if a mere linear string is postulated all the elements of which are of the same form.

Thus the only course left open to us is either to admit  $d = 0$  (and therewith infinite signal velocity) or to postulate arbitrarily exact delay elements. If we suppose delay members to be physical assemblies with spatial extension, then the possibility must be granted that such elements can be constructed in such a way that, coupled back into themselves as 'clocks', they will after an arbitrarily long time have readings differing by at most one time pulse. We deny this possibility to an even greater extent than does relativity theory, for we shall deny it even for clocks at rest relative to each other. We can now deliver this as a postulate:

P1 *The synchrony of clocks can be brought about only by means of communication.*

A more far-reaching formulation of this postulate is of some epistemological interest: The concept of 'the synchrony of two clocks' receives its content exclusively through the description of a closed signal chain that contains both clocks.

In this formulation the postulate is not to be defended at this

time; on the other hand, however, we do accept the first form and we shall substantiate this with the postulate regarding the finiteness of the density of information, for we can describe an experiment in which the pulse times of the two clocks are compared with any desired exactness by counting pulses and in which, therefore,  $\Delta t_1$  is measured with reference to a unit  $\Delta t_0$ . Any positive rational number can be obtained as the measure of  $\Delta t_1$ . If we assume that the sequence of these numbers will, as the experiment progresses, approach a limit value that is dependent only on the physical state of the clocks at the beginning of the experiment, then we ascribe to the pair of clocks an information content that exceeds any bound. Since we do not possess, let alone manipulate, this information content, it follows that it is conceptually simpler to forego even mentioning it in the construction of a theory of communication. This important step will be carried out in all cases involving real numbers: spatial coordinates, physical measure of a state (of which we shall speak explicitly), probability, et cetera.

### 3. Prolegomena to a Reformulation

By means of the arguments advanced above, it has been shown why we must reject not only switching elements without time requirements ( $\wedge$ ,  $\vee$ ,  $\sim$ , ...) but also those with fixed specified time requirements ( $\Delta t$ ) as idealizations of our elementary logical components, if we are to keep the physical realization of automata of variable total structure in sight. We may reject them also, however, for automata of fixed finite structure. It is to be shown now that it is possible to specify other idealizations for logical components such that the following is valid:

P2A Every logical element in its ideal-

ized form has a manipulable model  
in every field in which actual communication takes place.

If this is successful then mathematical-logical models of general communication processes can be constructed that are exact in a stronger sense than could be achieved without this assumption. Also it is to be hoped that such models will enable a deeper view into the nature of communication.

By *communication* we understand all forms of the actual flow of information. The information flow describes the structural properties of the quantities that are known as 'effect' in physics. The requirement P2A should assure us of the universality of our model of communication. But how should one demonstrate that the condition is fulfilled? Such a demonstration will not be attempted here, but we do wish to make such a demonstration as easy as possible; this we do by changing the requirement P2A from its indirect form into its direct form and, therewith, by strengthening it somewhat:

P2 In the interpretation of idealized logical elements, one may appeal only to such finite properties of the physical objects realizing these elements as have their invariance insured by a *general principle*.

The problem is therewith shifted to the concept of the general principle (GP). The philosophical implications of the concept need not concern us here. We wish to give three examples of GPs that are already sufficient for the switching elements that are to be introduced below (Section 5):

- GP1 The existence of a quantised quantity  
(e. g. effect, change)
- GP2 The validity of a law of conservation  
(e. g. of charge)
- GP3 The validity of a reaction principle  
(e. g. for forces)

These principles have been selected because they have the form of the best assured pronouncements of physics. It turns out that, taken together, these severely limit the freedom in the choice of logical elements. We must show, then, that from the elements yet permitted we may actually produce structures that can 'steer' the flow of information in any way that might be desired.

Now an obvious objection is that such a renunciation of the classical switching elements leads us directly into the field of asynchronous automata, which are regarded as mathematically so difficult that one must give up hopes for clarity (see [18], p.204), indeed so difficult that a practical application can be considered only in the simplest case. Remarkably enough this objection disappears completely if we construct the theory in a consistent way anew with the application of the postulates P1 and P2, therefore with the renunciation of certain additional, normally accepted possibilities, thus by limiting our means of representation. Further we seek a way in which to describe deterministic processes with the same means and in which to explain the difference between the two by means of the topological properties of finite structures over discrete sets.

An essential characteristic of the way in which we shall proceed is that we must completely reject metrical properties of all types. Our only

presupposition is that there are objects with some invariant properties (to be interpreted, according to the context, as energy amounts, effect quanta, electrical charges, material particles, etc.), between which there are bounded connexions. We shall connect these objects into nets and show that there exist nets equivalent to a Turing machine, the distant parts of which nets can be produced without reference to the starting point, thus without communication between arbitrarily distant distant parts of the net(s). It will thus be shown that such a net, even if it consists of only a few objects, can not be distinguished by an experiment from an 'infinite' net capable of recognizing recursive messages, and that, to that extent, such a net will be equivalent to the 'infinite' net.

In order that the connexion with applications might emerge more clearly, we shall forego the mathematically more elegant possibility of treating the temporal neighborhood relation and the spatial neighborhood relation between phase cells in the same manner; we shall therefore speak of nets and their modifications rather than of cell complexes.

We shall now proceed as follows:

- a) We specify a method of producing, with the aid of intuitively selected nonmetric switching elements, certain nets that reproduce the physical flow of information, thus the structural relationship of event classes. We therewith fulfill only GP1, not, however, GP2 or GP3 (Section 4).
- b) We limit the choice of switching elements by requiring that GP1, GP2, and GP3 be fulfilled, so

that P2 can be verified. The way is therewith cleared for applications, and the basic phenomena of communication are representable with purely combinatorial-topological means (Section 5).

At step a) we shall speak as if the discrete objects were embedded in a continuous space-time world, in order to make understanding somewhat easier. It should be remarked, however, that this is by no means necessary; the ostensible continuum rests in the final analysis on the erroneous assumption that the axiom of density (see [6], p.154) has an operational sense (see for example [20]). The axiom is rather a rhetorical instrument for the construction of hypotheses in the sense of inductive logic (see [5] or [21]).

#### 4. A Reformulation

Using the and-gate (Figure 2) as an example, we shall explain the approach that is to be introduced presently. Such a switching element connects three things (wires, for example) that assume certain states (potentials, for example) at certain times. This connexion is of the

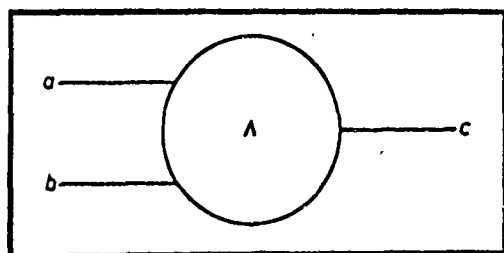
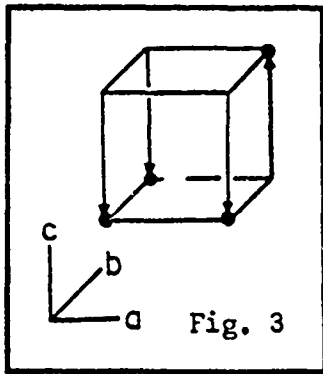


Figure 2.

following sort: let the sets of possible states of a, b, and c be  $S_a$ ,  $S_b$ , and  $S_c$  respectively; let  $O_a$  and  $I_a$  be disjoint subsets of  $S_a$ , and so similarly for b and c. The effect of the gate

is described by means of a mapping of the set of all state-triples into itself; the fixed points of the mapping are the stable state-triples. We can characterize the mapping by means of directed line segments con-





necting the vertices of a cube. Figure 3 characterizes the following mapping for the and-gate:

$$\begin{array}{l}
 0a \ 0b \ 0c \rightarrow 0a \ 0b \ 0c \quad . . . . . \\
 0a \ 0b \ 1c \rightarrow 0a \ 0b \ 0c \quad 1a \ 1b \ 0c \rightarrow 1a \ 1b \ 1c \\
 0a \ 1b \ 0c \rightarrow 0a \ 1b \ 0c \quad 1a \ 1b \ 1c \rightarrow 1a \ 1b \ 1c .
 \end{array}$$

The point designating the state of the gate passes through the weakly marked edges through the action of influences external to the circuit; the strongly marked edges it passes through 'of its own accord', that is to say, on the basis of the physical properties of the switching element. It should be remarked that in the traversal of these c-edges the state of a and b may not be changed in such a way that it again leaves the particular subset  $0a$ ,  $1a$ ,  $0b$ , or  $1b$  to which it belongs. That may be achieved with certainty, however, only by virtue of the fact that the c-edges run parallel to the c-axis of a state-coordinate system without regard to whether the state-sets are considered to be dense or discrete. That means, however, that a physical process must here run its course with no reaction on its cause. Thus, even without the requirement that the process take place 'in a very short time', we have no exact physical model available that behaves correspondingly: the and-gate does not satisfy postulate P2.

We shall now examine the switching element W (Figure 4), which has

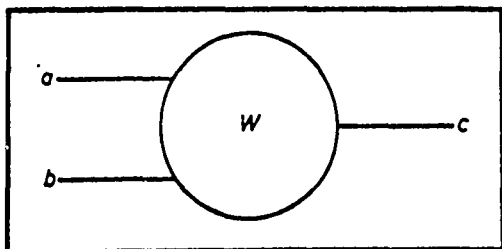


Figure 4.

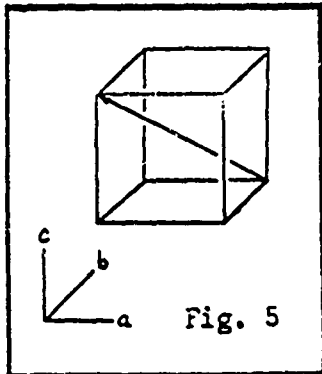
the following simple properties (see also Figure 5):

$$1a \ 1b \ 0c \rightarrow 0a \ 0b \ 1c$$

(all other triples are stable).

This switching element we could de-

signate as an 'and-gate with reaction': in that c enters into the state lc, a and b are shifted into the states 0a and 0b respectively. Since



all states are changed in the traversal of the 'active edge' of the element, the difficulty of interpretation mentioned above does not even arise. We can elucidate the behaviour of the switching element as follows: *W waits* until it observes the objects la and lb in its neighbor-

hood and produces from these a new object lc. This waiting of the switching element *W* should, however, by no means be thought of as an activity, say of the type such that the element repeatedly interrogates the states of a, b, and c and then on the occasion of some particular result of the interrogation changes the states of a, b, and c; rather it merely represents a connexion of a, b, and c that will occasion a state change only if la, lb, and 0c are all present, and the state change consists only in the fact that after its entrance (or occurrence) the objects 0a, 0b, and lc are present.

The process of the substitution of 0a, 0b, lc for la, lb, 0c we view as a single *elementary process* that can no longer be divided into separate parts; below (Section 5) we shall admit only physically meaningful processes as elementary, a procedure that first complicates the constructions considerably but yields remarkably simple results.

On the other hand there shall be required of a connexion (switching element) *W* nothing more than the described behaviour; we require only that 0a 0b lc be a (*temporal*) *successor* of la lb 0c, and we shall write

$$W (a b c) : 1 1 0 \rightarrow 0 0 1 .$$

We need say nothing regarding a 'time requirement' of this process. As must be clear from the interpretation, the disappearance of an object  $1a$ , the disappearance of an object  $1b$ , and the appearance of an object  $1c$  are all the same act.

For the characterization of the full breadth of the possibilities of interpretation the following remarks are essential:

1) Duality between 0-objects and 1-objects: We can characterize the same switching element  $W$  in two ways:

$$W \equiv 1a \ 1b \rightarrow 1c \quad \text{and} \quad W \equiv 0c \rightarrow 0a \ 0b.$$

If instead of 1-objects we observe the 0-objects dual to them, then the operation of  $W$  consists in the splitting of one object into two objects different from it and from each other. A net consisting of such elements and similar ones may therefore be conceived of in different way according to which of the two object classes one wishes to mark. We shall therefore make the following agreement: the assignment of the symbols 0 and 1 to the object classes is arbitrary but it must be kept fixed. If we wish to distinguish an object class we shall do this by assigning the symbol 1 to it. By interchange of the object classes a switching element passes over into its *dual*:

$$\bar{W} (a \ b \ c) : 0 \ 0 \ 1 \rightarrow 1 \ 1 \ 0 \ .$$

For switching elements that change all of the objects treated to another class, the passage to the dual element is obviously equivalent to an interchange in the sense of temporal direction.

2) Operation Domain of a Switching Element: The 'connexions' of a switching element, designated by  $a$ ,  $b$ , and  $c$ , are its (three) *domains of operation*: if a 1-object (or in the dual case a 0-object) appears in a

domain of operation it can be perceived by the switching element.

One possible way of interpreting the domain of operation is by assigning to it a spatial (or spatio-temporal) region. The question then arises as to the exact delimitation of this region, but we shall once again arrange our basic concepts in such a way that the question will have no object. We make the following agreement: if in an interpretation spatial (or spatio-temporal) regions are assigned to operation domains, then these regions shall not be capable of reinterpretation except as domains of operation of switching elements. The only things we need then are the topological properties of regions: separation, intersection, and inclusion.

Domains of operation with no statement of their interpretation we shall, for brevity, refer to as *locations* (or *places*). The switching element  $W$  is thus 'three-place'.

In the most important interpretations each domain of operation can take in only a uniform limited number of objects. It suffices then, as will be seen, to allow only locations that contain exactly one object at all times. The transition to 'locations' that at all times contain exactly  $n$  objects, which we can call *quasimetric locations*, is always possible by connecting ordinary locations with suitable switching elements.

Switching elements with no statement of interpretation we shall refer to as *nodes*.

We now wish to admit the following relations between locations: identity, vicinity, and separation ( $\equiv$  nonidentity  $\wedge$  nonvicinity), but not general intersection or inclusion. We do this by means of the following definitions, in accordance with our previous agreement concern-

ing the treatment of the interpretation.

If  $a, b, \dots$  are the domains of operation of a switching element  $K$ , then we say that the locations  $a, b, \dots$  *belong to* the node  $K$ :

$$J(a, K), \quad J(b, K), \quad \dots \quad (\textit{incidence}).$$

Two locations  $a$  and  $b$  are *identical* if they contain the same objects:  $I(a, b)$ . Thus it is immaterial whether we take identity between locations or between objects as basic. The relation  $I$  is reflexive, symmetric, and transitive; we ought to call the equivalence classes of locations that  $I$  imposes  $I$ -locations, but we wish, further on, to observe only these, and we wish to use the word 'location' for these, too. The distinction does not become essential until we completely formalize the rules for net construction.

Two different locations are called *vicinal*, and we write  $N(a, b)$ , if there exists a node to which both belong:

$$N(a, b) \equiv (\exists K)(J(a, K) \cdot J(b, K)) \cdot \sim I(a, b) \quad .$$

The relation  $N$  is irreflexive, symmetric, and nontransitive.

We observe only those locations that belong to at least one node, for it is only through incidence with a node that a location can take part in communication:

$$Lo(a) \equiv (\exists K)(J(a, K)) \quad (a \text{ is a } \textit{location}) \quad .$$

Two different locations are called *connected*, and we write  $C(a, b)$ , if there is a natural number  $n \geq 1$  such that there is an  $N$ -chain of  $n$  links between them:

$$C(a, b) \equiv (\exists n)(N^n(a, b)) \quad .$$

The relation  $C$  is symmetric and transitive.

We now need a set of two elements, which we designate by the sym-

bols 0 and 1. We shall call these elements *bits* and write:

$$\text{Bit } (i) \equiv i \in \{0, 1\} .$$

We shall define an *object* as an ordered pair consisting of a location and a bit and we shall write  $\text{Ob } (a, i)$ :

$$\text{Ob } (a, i) \supset \text{Lo } (a) \cdot \text{Bit } (i) .$$

Thus, we shall use the word 'object' here in such a way that it would be senseless to say that an object moves from place to place (location to location). The genidentity (see [6], pp.199f.) of information sets (or particles) must rather be described by the statement of a net structure; this genidentity is not necessarily independent of the observer.

The N-relation imparts a structure to a set of locations, the *N-structure*, and we shall call a connected N-structure an *N-net*. By the valuation of the locations of an N-net we obtain a connected N-structure over a set of objects. This structure represents the 'state of a system' in automata theory. There it is presupposed that one can state or even produce a successor relation over a set of such structures. Here we free ourselves of this condition by demonstrating that it cannot always be fulfilled.

We shall admit instead only the successor relation ' + ' for ordered  $p$ -tuples of such objects as (so to say) occur at locations belonging to a single node. For unlimited  $p$  we obtain once again the conceptual structures of the theory of synchronous automata, which would represent in this view a sort of theory of action at a distance. The essential deviation of our approach from that of the theory of synchronous automata lies then in the limiting of  $p$  (to small natural numbers), and in the interpretation only short-distance effects (actions) are taken into consideration.

As an initial attempt we shall arbitrarily choose

$$P = 4 \quad 1 \leq p \leq P$$

and we shall agree that in the statement of the *actions* of a node, e. g.

$$p = 3 : W (a b c) : 1 1 0 \rightarrow 0 0 1 \quad ,$$

we shall always name all of the locations belonging to the node. This is no limitation of generality. There are thus no more than  $P$  locations belonging to any node.

The *inverse* of an action derives from an interchange of the left and right sides of the successor relation. We shall call a node *reversible* if for each of its actions it also possesses its inverse. The number  $F$  of possible different switching functions, i. e. of examples of nodes with actions, is bounded, namely

$$F \leq \sum_{p=1}^P 2^{(2^{2^p})} < 2^{(2^{2^P+1})}$$

Thus for  $P = 4$ ,  $F < 2^{2^{57}}$ . We wish to make do with but a small number of these, however, and we define the following actions:

$$\begin{array}{lcl} Q (a) & : & 0 \rightarrow 1 \\ T (a b) & : & 1 0 \rightarrow 0 1 \\ W (a b c) & : & 1 1 0 \rightarrow 0 0 1 \\ V (a b c) & : & 1 0 0 \rightarrow 0 0 1 \\ & & 0 1 0 \rightarrow 0 0 1 \\ U (a b c) & : & 1 0 0 \rightarrow 0 1 1 \\ & & 1 1 0 \rightarrow 0 0 1 \\ S (a b c d) & : & 0 1 0 0 \rightarrow 0 0 0 1 \\ & & 1 1 0 0 \rightarrow 0 0 1 0 \end{array}$$

and the following inverses:

$$\begin{aligned} \bar{Q} (a) & : & 1 \rightarrow 0 \\ \bar{T} (a b) \equiv T (b a) & : & 0 1 \rightarrow 1 0 \\ \bar{W} (a b c) & : & 0 0 1 \rightarrow 1 1 0 \end{aligned}$$

These functions have been chosen only because of their clarity, and in Section 5 they will be replaced by simpler ones. Here we have sources (Q) of distinguished objects, transport (T) of bits, the waiting (W) for the appearance of two particular objects, the joining (V) of bit-channels, the dualization (U) of an object, and a function (S) that permits the destructive readout of a store. The defined relations can obtain only between vicinal locations; thus, for example,

$$S(a b c d) \supset N(a b) \cdot N(a c) \cdot N(a d) \cdot N(b c) \cdot N(b d) \cdot N(c d) .$$

There is also implied therewith the nonidentity of the locations of a node.

An *f-net* we shall define as a structure produced by elements from a set *f* of *P*-limited switching functions (i. e. switching functions connecting at most *P* locations) over a set of locations such that the implied *N*-structure is an *N-net* and is thus connected.

For the most important interpretations we must impose on the *f*-nets a further limitation by taking into consideration the fact that not every object is immediately accessible from all locations at all times. We require that no location belong to more than *M* different nodes:  $1 \leq m \leq M$ . We now consider the important case in which  $M = 2$ . By the term *net* we shall therefore mean an *f-net* with  $P = 4$  and  $M = 2$ .

The *N*-structure of a net can be conveniently represented then by means of a graph whose edges and nodes correspond, respectively, to locations and nodes. We now wish to arrange certain nets in such a way that



for every object it is always uniquely determined in which action of which node it can take part. The behaviour of the net is then obviously determined if we require additionally that different actions of a node differ in their left-hand sides.

By a (*case of*) *conflict* we shall mean a situation in which an object belongs to the left-hand side of more than one applicable action. It is then completely indeterminate which of the actions is applied. The following is criterion for the identification of a conflict: A conflict is said to have been present if and only if an applicable action loses its applicability other than through its application. Nets in which conflicts are impossible will be said to be *conflict-free*. This freedom from conflict must be proven by means of its 'hereditary character' viz-a-vis the steps in the construction of the net.

But what is to happen to the critical object in the case of conflict? In consideration of the interpretations we propose the following:

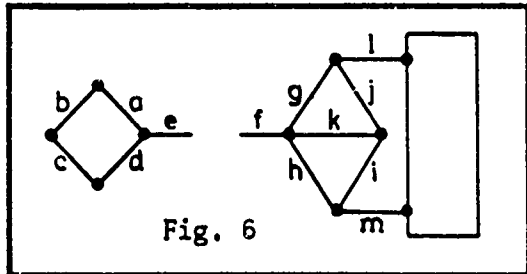
Axiom: In case of conflict at most  
one action can take place.

The only thing that remains undetermined (or indeterminate) then is which of the actions comes into question; in no case can several actions be applied simultaneously.

The *peripheral locations* of a net we shall define as those locations that belong to exactly one node of the net.

If we identify a pair of peripheral locations of two conflict-free nets, it is not necessarily true that a conflict-free net will result. The same applies if the same nodes are connected by T-nodes instead of being identified. This situation may be seen in the following example.

We may take the following as an example two conflict-free nets being combined to form a net that is not conflict-free. In Figure 6 let Net 1 be a 'clock', Net 2 an observer with his own clock; we define



Net 1: T (a b)	Net 2: S (f k g h)
T (b c)	$\bar{W}$ (l j g)
T (c d)	$\bar{W}$ (m i h)
U (d e a)	V (i j k)

Initially we let the following conditions prevail:  $la, lk$ ; we let the rest of Net 2 act as  $\bar{Q}(l)$  and  $\bar{Q}(m)$ . Each net by itself is conflict-free, but the measurement of the 'motion' of the clock a b c d with reference to the 'motion' of clock h i k by means of the connection or identification of e and f is always bound up with the possibility of a conflict the critical object of which can lie in either e or f. These circumstances indicate to us a close connection between measurement and conflict that will not go unattended.

A conflict-free net with two peripheral locations that contains only  $n > 1$  T-nodes we shall call a *channel*. The T-nodes must be oriented in the same way, and there is given therewith also an orientation of the channel. An *oriented* location is defined as a location that belongs to exactly two nodes such that all the actions of the first node carry the object of the location over into its dual and all actions of the second node do the same in the opposite sense. In the graph representation we shall designate such locations by a directed arc.

A net is said to be *capable of communication* if it has peripheral locations. Communication between two nets is made possible by the identification of peripheral locations. There arises then an aggregate net that

can be described with the same methods as the individual nets. When we speak of communication *with* a net we imagine ourselves in the position of a partial net, but we do not immediately have the possibility of applying to this partial net the same forms of description as were arranged for nets. Rather we must use colloquial expressions for the actions that we exert on the common peripheral locations. Thus there will appear such concepts as the following: 'signal', 'information', 'condition', 'until', and various others.

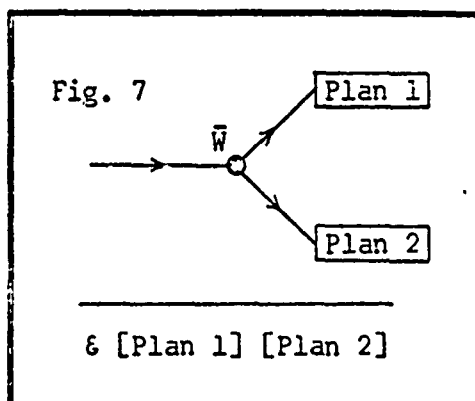
On the other hand we can frequently specify a net that exactly (or at least to some exactly delimitable extent) describes our behaviour, our intentions, our cognizance, etc. The comparison of the colloquial formulation and the net will thus yield a sort of prescription for translation between colloquial language and net language, which except for the explicit and partially divergent treatment of temporal connections is similar to the language (or linguistic forms) found in symbolic logic (see [6] for example).

It has turned out, however, that a large number of nets that can be obtained in this way are not conflict-free, and it can often be demonstrated that there exists no equivalent conflict-free net, as in the following example: We follow a *plan* of the form 'If A or B, then C'. Let  $NA$ ,  $NB$ , and  $NC$  be any nets with which we can communicate. It should be emphasized that communication can take place only via the peripheral locations; if we were to allow the possibility of changing the nets under our control by operating on interior locations (i. e. nonperipheral locations) or interior nodes, then we should reach a contradiction of our definition.

The condition 'if A' means in the interpretation of the plan that in order to test whether the condition is fulfilled we give a signal to NA; that is to say, by means of an action we alter the object of a peripheral location of NA and we make note of the fact that we have changed it. We now begin to observe the peripheral locations of NA (or perhaps the peripheral location with changed object as the only one) and prepare an action that becomes applicable if and only if NA carries out certain actions on its peripheral locations thereby changing the peripheral objects. We can even prepare two actions, which will be assigned to a 'yes' or 'no' answer from NA. The execution of a prepared action represents the perception of the answer. Even if we have exact knowledge of the structure of NA, we can in no way force the answer within a prescribed 'span of time.' This mode of expression would have the following meaning: we have a cyclic chain of action that is not in connexion with NA; we repeatedly interrogate the peripheral objects of NA 'to the beat' of this clock. The possibility of interrogation means, however, that the chain of actions proceeding through NA to the answer is likewise closed; it thus represents a second independent clock. However, according to the postulate P1, the synchrony of these clocks can be effected only if the passages through the cycles are causally (conflict-freely) dependent on each other. Thus if we recognize P1, then varying according to the particular *modus observandi*, either the word 'interrogate' or the term 'span of time' will lose its meaning in this reference.

The same considerations would apply for communication with NB.

'Then C' would mean, then, that (on fulfillment of the total condition 'If A or B') a signal is given at C. Possible feedback signals from



NC may well belong to other parts of a total plan; these other parts would be connected to the observed plan by means of a *parallelity operator* ' $\&$ ' signifying that not only is the observed plan to be executed but the totality of remaining

partial plans as well. The *activation* of such composite plans is brought about by way of a node  $\bar{W}$ .

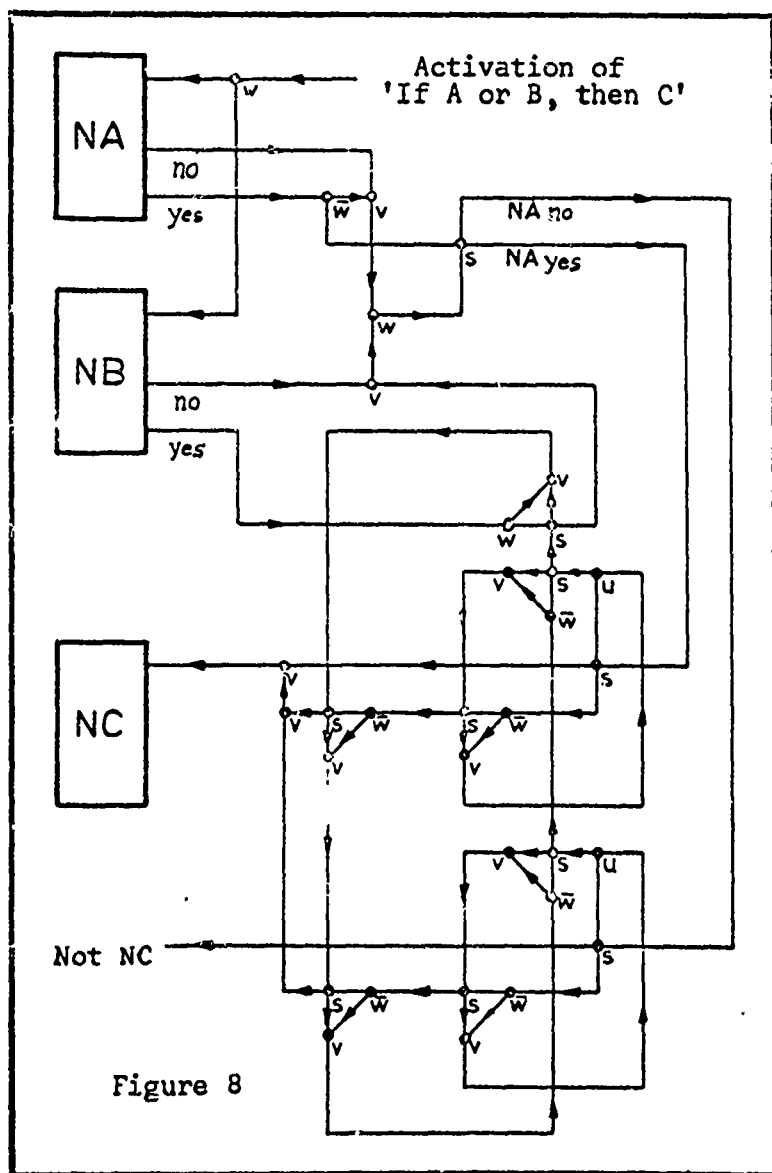
We must now analyze the plan 'If A or B, then C'. This analysis will consist in finding a net representation of the (colloquial) 'or' from which an exact operational definition of 'or' can be had. This functor does not in general correspond to the disjunction of the two-valued predicate calculus, which (as is the case for all of the functors of this calculus) can be represented conflict-freely.

Below (Figure 8) we give a conflict-free representation of disjunction. We have not chosen the 'simplest' representation (as regards the number of nodes, actions, or locations), but rather we give one from which it may be seen how for any given predicate function a net can be constructed. In essence the net says the following: After the activation of the total plan both NA and NB are activated. The partial information obtained through the interrogation is stored, and ready-signals are derived from the process of storage; when both ready-signals (one derived from the NA interrogation, the other to the NB interrogation) are present, the 'items' of stored information are coupled according to the truth-table for disjunction and read out.

Thus, for example, although the presence of a 'yes' answer from NA

alone defines the answer and would entitle us to the decision yes-NC, this net must nevertheless wait for the answer from the interrogation of NB. This corresponds exactly to the formal prescriptions for the evaluation of a truth table when its inputs are assigned directly to the possible answers (from NA and NB).

The noticeable complexity of the net is explained by our selection of switching elements that are subject to no more than two actions (each). Jumping ahead somewhat (see Section 5), we mention the fact that this complexity can be circumvented by the selection of a single switching element



and that this choice coincides with the requirements of postulate P2. We emphasize once again the arbitrariness of all constructions in this section; the constructions are meant not as suggestions for circuits, but rather as temporary means of proof.

A general remark for the understanding of net constructions: The representation of nets, as in Figure 8 and in what follows, is meant as an abbreviated form of a proof for the existence of a derivation of certain forms of communication from simpler ones produced by means of axioms and definitions, i. e. from actions. Corresponding to the elementary character of the relations represented, a formal elaboration of such a proof would be extremely voluminous. Similarly, even a fairly exact colloquial explanation would be too long to be clear.

On the other hand, however, we do not wish to appeal to theorems with universal quantifiers, in order that the proofs be verifiable in the strictest constructive sense. We shall therefore content ourselves with the statement of a method of not only producing full proofs but also of attaining to some insight into the workings of a net.

We imagine that the execution of the actions, be under our control, but not in such a way that the actions depend on additional locations, but rather as if we were to assume the rôle of the node ourselves, so to say. We then enumerate all the sequences of possible actions of the net and determine their dependence on our own actions; this we do for each node of the finite net.

The following 'operational' representation corresponds to certain interpretations of the many-valued predicate calculi, for example to the following truth table for 'or':

A		T	T	T	U	U	U	F	F	F
B		T	U	F	T	U	F	T	U	F
A or B		T	T	T	T	U	U	T	U	F

We interpret the inputs to the truth table as follows:

T: the result of a given experiment is *a* ('yes')

F: the result of a given experiment is *b* ('no')

(where *a* and *b* are mutually exclusive)

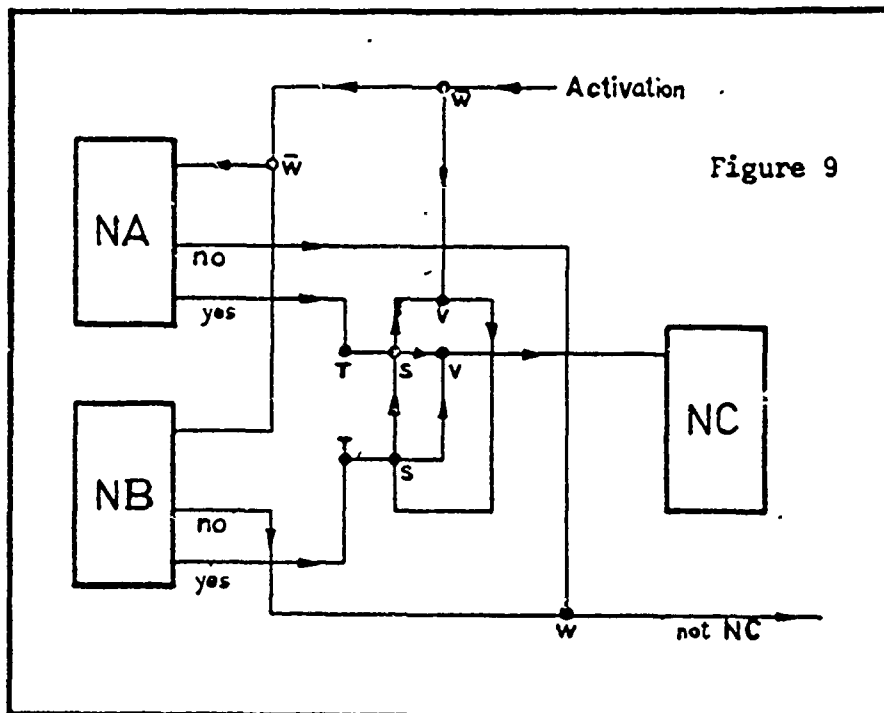
U: the result of a given experiment is unknown.

In order to avoid complications, we assume that the total plan provides for only a single evaluation of the condition 'If A or B'. Since it must be decided by some node whether it is A or B or both that represents the cause of the activation of C, the net evaluating the condition will nevertheless contain a conflict. By way of an example, if the action of making A alone the cause of C has been enabled but not yet executed, there is nothing to prevent B from appearing before the action takes place. Therewith, either a further action at the same location is enabled or else the first action is made inapplicable before execution; in either case there is a conflict.

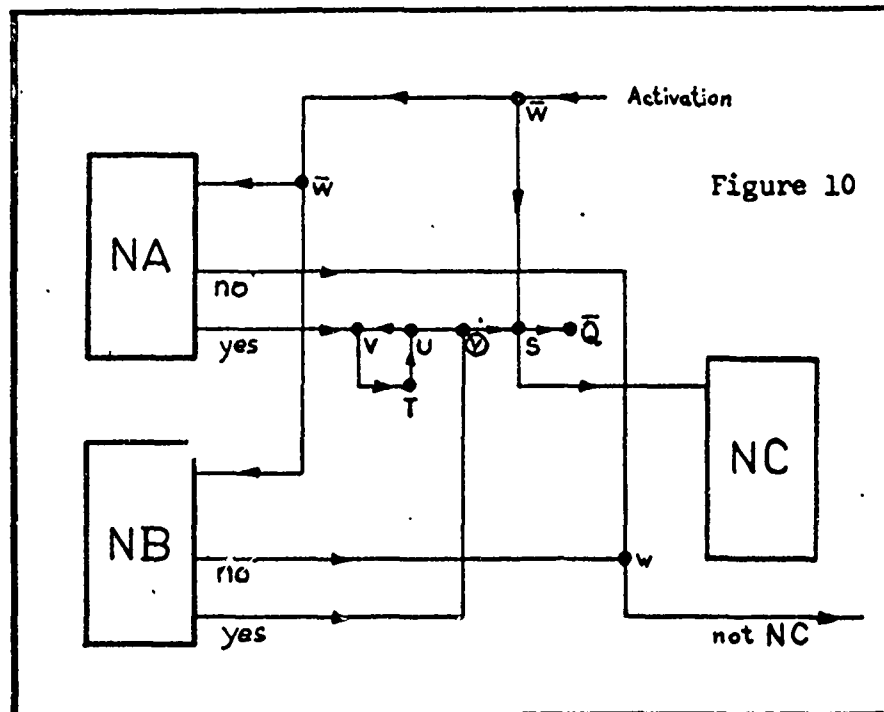
The nets below (Figures 9 and 10) reproduce the above truth table, but they also say somewhat more in that they, in characteristically different ways, distinguish more exactly the locations of possible conflict, i. e. they give two processes for the evaluation of the truth table.

In the first case (Figure 9) NA and NB have (speaking quite colloquially) equal say in the evaluation. The critical objects of the two possible conflicts lie on the  $\overset{\bullet}{T} \text{---} \overset{\bullet}{S}$  locations. The conflicts involve the actions of the S-nodes.





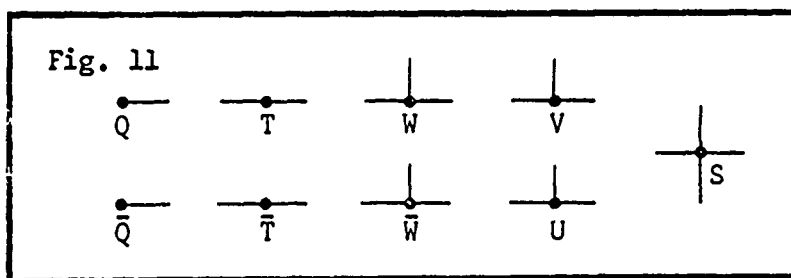
In the second case the critical objects lie at the input locations of the encircled V-node,  $\textcircled{V}$ . Here there is a priority of an NB-yes answer and the conflicts involve the actions of  $\textcircled{V}$  and U. The S-node is to insure that NC be activated no more than once.



Two things should be clear from these examples: 1) the necessity of admitting conflicts in the exact representation of certain forms of communication, and 2) the difficulties encountered from the beginning in the construction of conflict-free nets.

We now wish to construct a conflict-free net that will be capable of communication and will correspond to a Turing machine. We plan the construction in such a way that that it will at the same time become clear how one can design conflict-free nets 1) for arbitrary switching mechanisms of fixed finite size, and 2) for arbitrarily extensible memories. Here, too, we shall be concerned less with the number of nodes or locations than with generalizability.

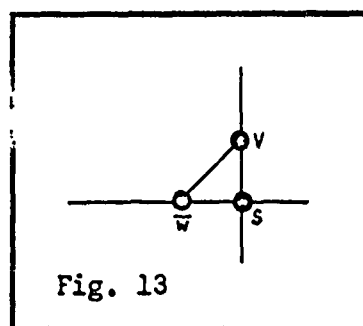
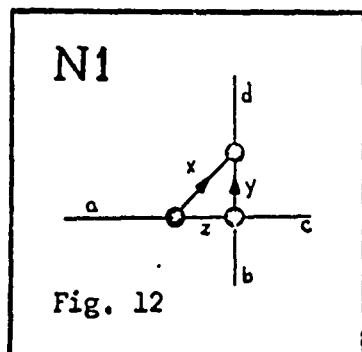
We have at our disposal the node types of Figure 11; using these we



shall define a new 'unit' N1 (Figure 12) as follows:

$$\begin{aligned}
 N1(a\ b\ c\ d) &= \bar{W}(x\ z\ a) \\
 &V(x\ y\ d) \\
 &S(z\ b\ c\ y)
 \end{aligned}$$

N1, abbreviated as shown in Figure 13, is used primarily as a part of N2,



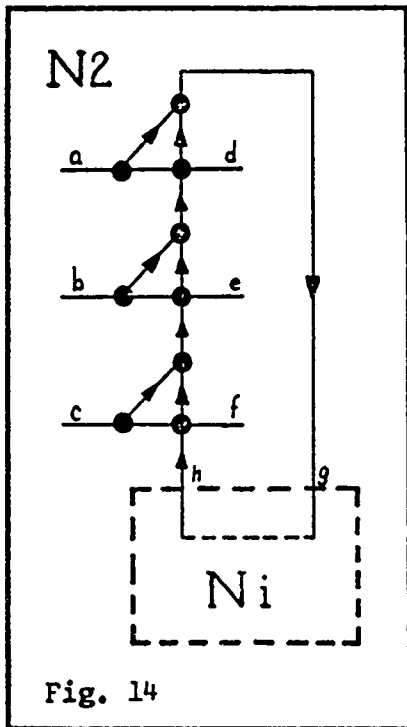


Fig. 14

shown in Figure 14. N2 can serve to produce the temporary connection of several action chains (for example, for activating a net Ni from various locations, as in Figure 14). It is required, however, that at any given instant at most one of the action chains be active; this condition is to be fulfilled by the environment of N2. Such conditions we shall express by means of formulas of a combinatorial system CS3, from which the concept 'form of communication' will receive an exact

meaning. Combinatorial system CS3 we define as follows:

Axiom:  $\Sigma$

Productions:  $\Sigma \rightarrow \vee \Sigma \Sigma$  (choice or conflict)

$\Sigma \rightarrow / \Sigma \Sigma$  (causality)

$\Sigma \rightarrow \& \Sigma \Sigma$  (parallelity)

$\Sigma \rightarrow * \Sigma$  (iteration, signal cycle; 'clock')

$\Sigma \rightarrow \emptyset$  (contradiction; insufficiency of the net)

$\Sigma \rightarrow a$   
 $\Sigma \rightarrow b$   
 $\vdots$

} enumeration of the elements of X

The CS3 formulas are to describe the temporal and structural 'connection' of a net with its environment (to the extent that this connection can be expressed by events in the peripheral locations alone). This connection we shall call a *form of communication* for the location concerned. By the term *event* we shall mean the appearance of a distinguished object. X we shall define to be the set containing all and only the possible peripheral events of a net.

We make the requirement initially (with no essential loss of generality) that the observed peripheral locations become, by virtue of the communication, oriented locations. The peripheral events then separate into two non-overlapping classes: input events and output events (dual interchange possible).

We shall forego any rambling colloquial explanation of the interpretation of CS3 because it is analogous to that of CS1 with the difference that communication forms can now be not only sequences of events

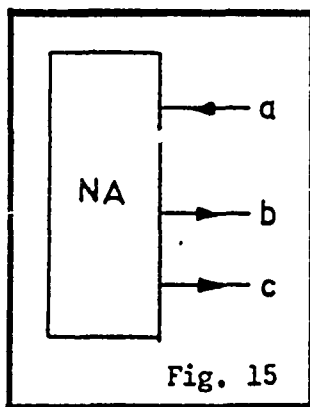


Fig. 15

but also *combinations* of events (because of the parallelity operator). We shall explain by means of an example:  $* / a \vee b c$  (as in Figure 15) specifies the communication form (*C-form*) of an information source. To keep to this C-form is to give the environmental net a structure that makes it im-

possible, after one interrogation to (of) NA, to give yet another before the answer to the first has appeared; we require further that NA give no answer without being interrogated, and that it give exactly one answer per interrogation and not both answers 'simultaneously' or 'serially' to the same interrogation.

$* a$  is a C-form of  $Q$  and  $\bar{Q}$

$* / a b$  is a C-form of  $T$  and  $\bar{T}$

$* / \& a b c$  is a C-form of  $W$  (and of  $\bar{W}$ )

$* / a \& b c$  is a C-form of  $\bar{W}$  (and of  $W$ )

$* \vee / / a d / b c * / b d$  is a C-form of  $N1$

C1:  $* / \vee \vee a b c \vee \vee d e f$  is a C-form of  $(N2 + Ni)$

C2:  $* \vee \vee / a d / b e / c f$  is a C-form of  $(N2 + Ni)$

Obviously C2 says more than C1; if we are familiar with C2 then we can derive C1 from it, but the formalism of the derivation lies beyond the limits set for this work. We would mention only that via the formulas of CS3 there can be defined a non-symmetric, reflexive, transitive relation of 'being contained in', as here:  $C2 \supseteq C1$ . We shall call C2 the *narrower* C-form, C1 the *broader*; here we shall not investigate the algebra of C-forms any further.

We wish to construct a net that has the following C-form:

$$C3: * / \nabla / a b / * / c d / a b \nabla / a b / * / c e / a b ,$$

and therewith the broader C-form:

$$* \nabla / a b / c \nabla d e .$$

The narrower C-form says in words that the net to be constructed has a

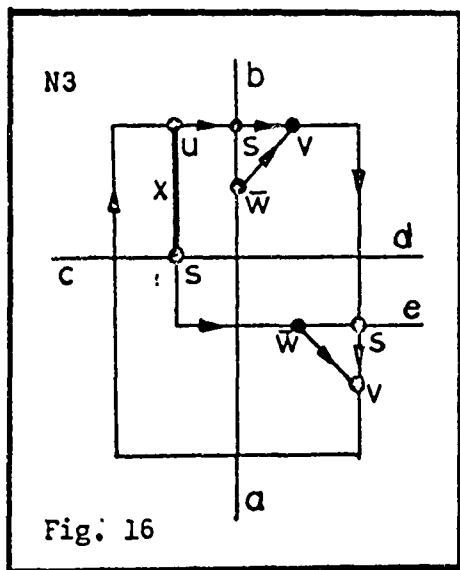


Fig. 16

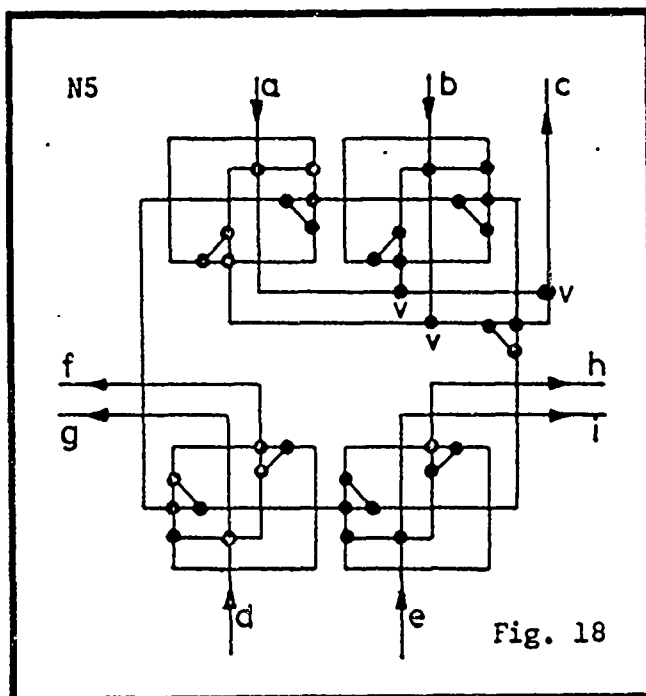
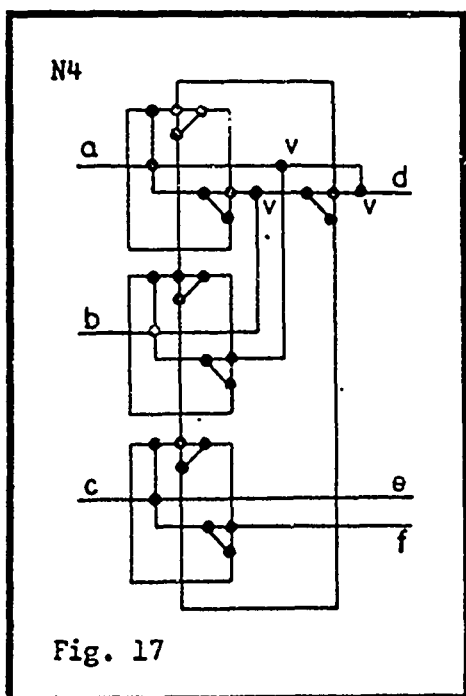
one-bit memory cell that can be interrogated by activating  $c$ ; the result of an interrogation is  $d$  if the total number of passes through  $/ a b$  is even,  $e$  if it is odd. The net N3 (Figure 16) does what is required. There it is to be recognized clearly that the concept 'state of N3', which is characterized by the object at the location marked 'x', is bound to the

communication form under which we observe the net.

We also construct another type of memory cell, N4 (shown in Figure 17), which can be set without counting the passes (mod 2) through  $/ a b$ ; this net has the broad C-form

$$* \nabla / \nabla a b d / c \nabla e f .$$

(C-forms in which every peripheral event is named exactly once may be called *normal*.) Since the net consists of a combination of tokens of N1, N2, and N3, we specify in the diagram only the additional nodes. A C-form corresponding to C3 that would make  $\nabla e f$  explicit can be found using the same considerations as in the case of C3. We shall make use only of normal C-forms henceforth.



Further we have the net types N5 (Figure 18) and N6 (Figure 19), which allow respectively for the interrogation and setting of memory contents from several (here 2) signal cycles. N5 has the normal C-form:

$$* \nabla \nabla / \nabla a b c / d \nabla f g / e \nabla h i$$

Setting    Readout 1    Readout 2

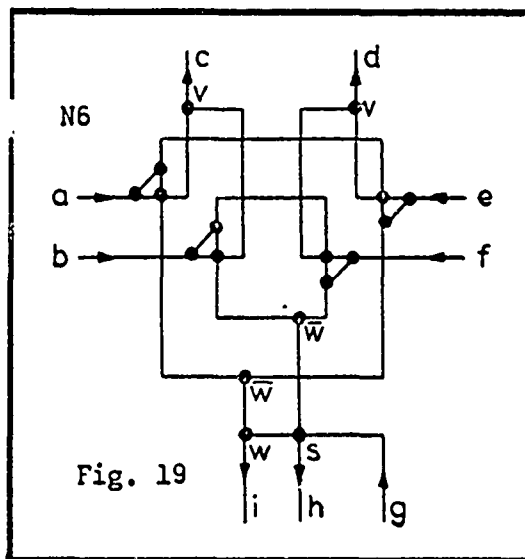
Here we have already oriented the peripheral locations in the diagram, although, strictly speaking, they receive their orientation only when the net is built into a larger net; we have done this to increase understanding and to express the fact that the C-form under which we observe the net requires this orientation.

The memory net N6 (Figure 19) has rather the character of a one-bit buffer, as follows from the C-form:

$$* / \nabla / \nabla \underbrace{a b c} / \nabla \underbrace{e f d} / g \nabla h i$$

Setting 1    Setting 2    Readout

Setting (from either the right or the left) and interrogation must take place alternately, beginning with a setting. Clearly, however, we can also build memories after the pattern of net N4 that are not subject to this limitation.



In order to put a general automaton into net form, then, we must first 1) establish a suitable communication form, and 2) select one of the various forms used for the representation of automata. As is known, it is no limitation of generality to admit as input and output of an information processor only alphabets with two symbol types instead of arbitrary finite alphabets. We assign to the symbol types the four peripheral locations of a net (*a*, *b*, *c*, and *d*). The timing of the automaton as well as other arrangements as to the way in which the automaton functions can be expressed by the normal C-form:

$$C7: * / \nabla a b \nabla c d .$$

The development of the C-form after the  $\nabla$  leads to a narrower C-form that reproduces the behaviour of a special automaton.

As the defining representation of an automaton we choose the state-transition graph or the table belonging to it (which may be incompletely filled-in).

$q_1$	$a$	$i_{11}$	$q_{11}$
$q_1$	$b$	$i_{12}$	$q_{12}$
$q_2$	$a$	$i_{21}$	$q_{21}$
$\vdots$			
$q_n$	$b$	$i_{n2}$	$q_{n2}$

with  $i_{jk} = \nabla c d$

$q_{jk} = q_l$

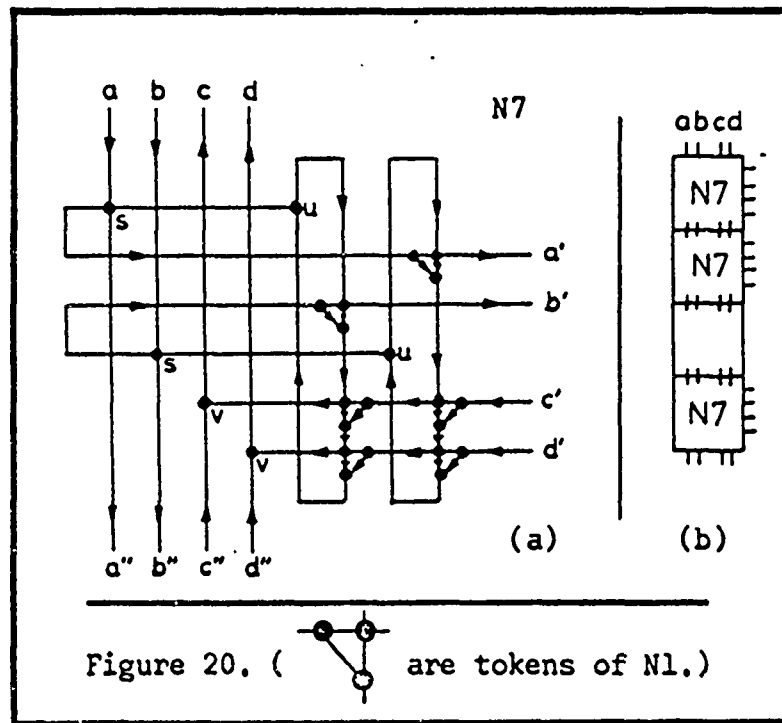
$(1 \leq l \leq n)$

We give the interpretation of an arbitrary line  $q_m \ b \ c \ q_l$  as follows: If the automaton is in state  $q_m$  and at the same time receives the input signal  $b$ , then it emits the signal  $c$  and goes over into state  $q_l$  for the next time period.

For each symbol type  $q_m$  that appears in the table (not for each symbol token  $q_m$ ) we introduce a copy of net N7 (Figure 20). We connect these copies of N7 with each other in the sequence in which the  $q_m$ 's are initially encountered in reading the table, and as soon as we would specify that there is to be no further extension of the net we 'close' the last peripheral locations of the type  $a''$ ,  $b''$ ,  $c''$ , and  $d''$  by means of  $\bar{Q}$ -nodes. The meaning of this closure is primarily 'denial of accessibility', that is of possibilities of communication. This closure is not merely a formal act with the purpose of observing certain arbitrary conventions regarding our constructions, for, obviously, by the execution of this closure, peripheral locations of the 'constructing subject' are freed, and the subject therewith attains to new, nonredundant communication possibilities.



The full gravity of such considerations is, of course, felt only in the complete formalization of the construction process; we present them here only in order to avoid misunderstanding regarding the meaning of closure.



By the process of coupling there arises an N7-chain the members (or 'links') of which are uniquely assigned to the  $q_m$ 's in the table. The particular  $q_m$  assigned to a given N7-link will be called the *mark* of that link. We connect the N7-chain to the automaton after the following prescription. For each line of the table (for example, the line reading  $q_m \ b \ c \ q_l$ ) we produce exactly one V-node and connect it as follows: the one input of the V-node is identified with the  $b'$  output of the N7-net that has the mark  $q_m$ ; the second input remains free; the output of the V-node is identified with the  $c$  input of the N7-net that has the mark  $q_l$ , or (if that input is no longer free) with the free input of the last V-node to be attached to the  $c$  input or to the V-node-chain of that input. And so *mutatis mutandis* for  $q_m \ a \ c \ q_l$ ,  $q_m \ a \ d \ q_l$ , and  $q_m \ b \ d \ q_l$ . After treating the last line of the table we close all peripheral loca-

... and  $d$  by means of  $\bar{D}$ -nodes

Above it was presupposed that any given symbol pair  $q_m i$  (for present state and present input, respectively) would appear in the table at most once (as is generally required of automata). We can free ourselves of this prerequisite, however, by replacing the above connection prescription with a matrix prescription, a prescription so simple that we shall give only an outline of it. Next to the N7-chain (of  $n$  links) we lay a  $2n \times 2n$  'matrix' of locations formed of  $2 \cdot (2n)^2$  locations that are to be oriented (see Figure 21). The row inputs we assign to the  $a'$  and  $b'$  outputs of the N7-chain; the column outputs we assign to the  $c'$  and  $d'$  inputs. Now for each row we introduce exactly one  $\bar{W}$ -node and one  $V$ -node as shown in Figure 22. After treating the last line of the table we close the 'gaps' between vicinal peripheral locations in the interior by means of T-nodes; the remaining peripheral locations we close by means of  $\bar{Q}$ -nodes. With this sort of coupling conflicts can arise, some corresponding to 'sneak paths' of relay engineering, but others of a more general type. On the other hand, even if the tables do not fulfill the above prerequisite, under certain circumstances, nets can arise that are conflict-free under the C-form C7. In order that fact can be judged correctly we mention without proof that for every net capable of communication there exists a C-form that permits conflicts in communication.

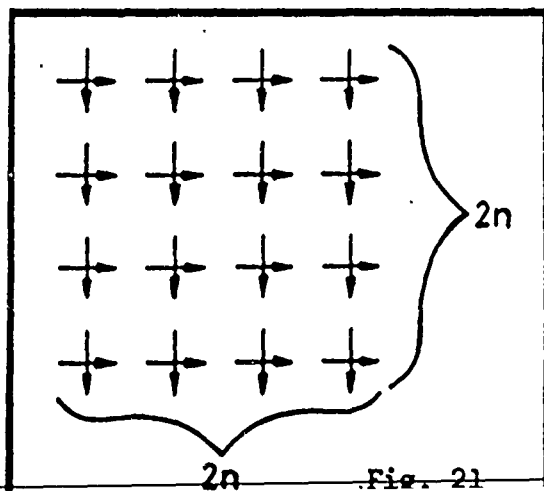


Fig. 21

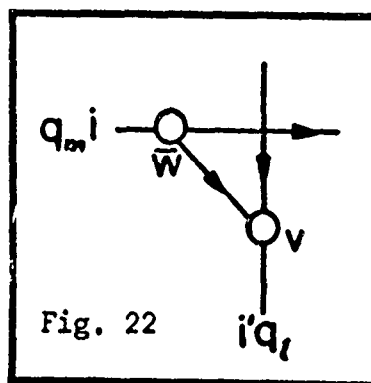


Fig. 22

We shall take this matrix representation of the partial nets of an automaton as basic in what follows (even in the construction of a net for a universal Turing machine capable of communication) with the one difference that not all of the peripheral locations that were closed with  $\bar{Q}$ -nodes will be so closed. In this way we shall be able to communicate with the machine via the peripheral locations of the matrix (and even in irregular languages, at that, and this, at our discretion, with or without conflict).

We shall not concern ourselves with the fixed (built-in) universal program, for, as is known, it can be put in the form of a finite table. We shall imagine this table to be supplemented by fixed standard programs (in the programming language of Turing machines, for input, output, tape erasure, and state interrogation - see [1]) and put into matrix form. There remains, then, the task of putting the *modus operandi* of the reading head and the tape into net form.

Here it is no longer a question of organizing for the solution of a total problem a prespecified number of elements capable of actions; rather we must find a form of organization in which unboundedly many elements can participate, but one that can nevertheless be described with finite means and of such type that each element communicate with only a uniformly limited number of immediately vicinal elements (which are always the same ones) with uniformly limited means.

We can prove the existence of such an organization form only by construction. Since the 'periphery' of the reading head with the tape is to communicate with the locations  $a$ ,  $b$ ,  $c$ , and  $d$  of an automaton, the head must have the C-form  $\ast / \nabla c d \nabla a b$  (designating the locations

to be identified with each other by means of the same symbol). We first ask ourselves whether this form is sufficient.  $\forall a b$  is sufficient in all cases for the symbols to be read from the tape together with their temporal delimitation; it is known that the tape of a Turing machine requires only two symbol types ('blank' and 'marked', or 'zero' and 'one'). We must also map the following symbols onto  $\forall c d$  :

$l$ : read the next symbol to the left  
 $r$ : read the next symbol to the right  
 $z$ : replace the symbol being read by 'zero'  
 $e$ : replace the symbol being read by 'one' .

Such a mapping is possible as follows: since on the basis of the construction of net N7 it is known at all times what the last symbol read was, we can combine  $z$  and  $e$  into  $u$  and  $w$ :

$u$ : leave the symbol being read unchanged  
 $w$ : replace the symbol being read by its dual.

We shall define:

$c \equiv l$  , and  $d \equiv / w r$  ,

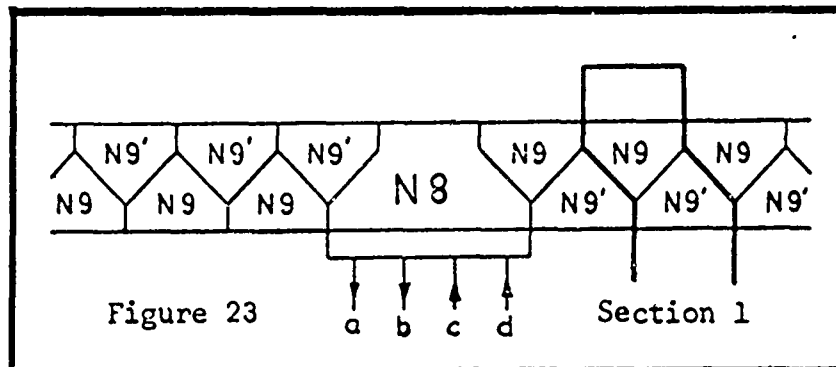
so that we thus have the following subprogram:

$l = c$   
 $r = / / d c d$   
 $u = / / / d c d c$   
 $w = / d c$  .

The programming language for Turing machines is therewith translated into the language of automata, and the four-place normal C-form  $* / \forall c d \forall a b$  is proven sufficient.

Above we have already seen descriptions for a one-bit memory (N5) that could be interrogated by different signal chains, and for a suitable one-bit buffer (N6); we shall now make use of these. (It is easily seen from the construction how one might design an  $n$ -bit memory or buffer.)

Our plan, then, is to translate the reading head together with the tape field being read into a net N8, and each of the two remaining 'halves' of the tape into a net N9 (as in Figure 23). If we imagine N8



to be removed, then there remain two memories of a special type, the bounded forms of which are well-known in programming, namely push-down (PD) stores (or automata). As in any general memory, so also in the PD store one has the following communication plan:

$$C11: * \nabla / c \nabla d e / \nabla h i g ,$$

in which  $\nabla h i$  is the information to be stored,  $c$  the 'fetch' signal for the last information stored,  $\nabla d e$  the answer to the interrogation  $c$ , and  $g$  the ready signal.

However, according to the results of Section 2, it is *impossible* for there to be such a simple communication form ruling in all of the completely separate sections (as 'Section 1' in Figure 23); otherwise we could interpret the C-form as a CS1 formula, and we should have to show that there exists a finite CS1 formula that reproduces the PD-principle, which is obviously recursive. It is therefore *necessary* for the C-forms of the sections to contain a noneliminable parallelity operator.

There are many possibilities for the intuitive selection of C-forms for the sections; their transduction can be effected with the aid of a calculus that we shall not describe here. We shall state a C-form direct-

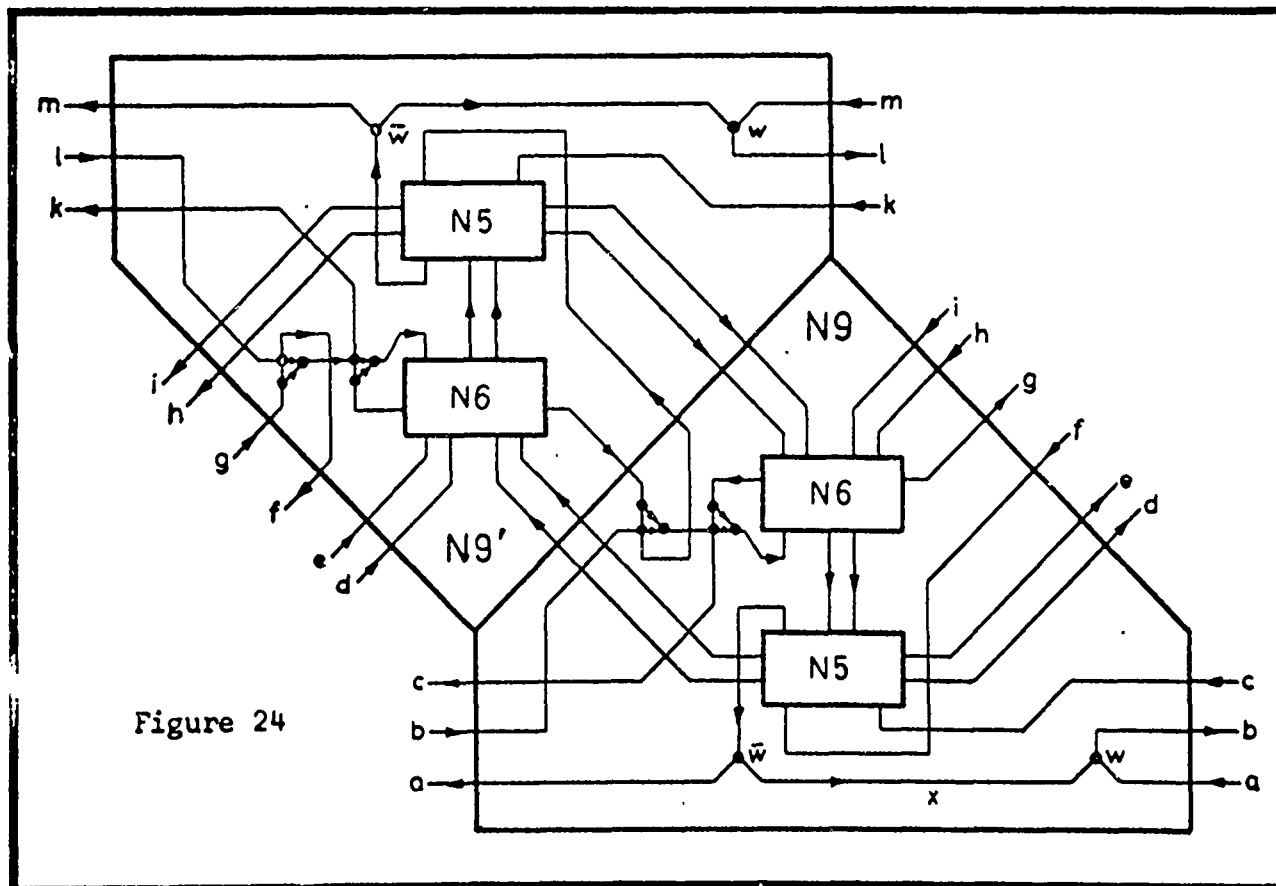
ly that contains only a single example of the  $\xi$ -operator and therefore illustrates quite clearly the noneliminability of the  $\xi$ :

C9:  $\ast / \xi \nabla / / / / c \nabla d e m \ell k / / / / \nabla h i g m \ell f a b .$

In contradistinction to corresponding 'synchronous' constructions, this C-form makes it possible for an action chain of uniformly limited length to proceed to the net periphery from each activation of  $c$  or  $\nabla h i$ , with no dependence on the length of the push-down store. In Figure 24 we specify a net N9 that corresponds to C9; N9' is produced from N9 by reflection (not by rotation). Each N9-N9' pair of nets communicating with each other satisfies the specified C-form conflict-freely with respect to both 12-tuples of peripheral locations. Proceeding from the periphery of such a push-down store, we can separate the C-form

C10:  $\ast \nabla / c \nabla d' e' / \nabla h i g'$

from the C-form C9 by means of the net specified in Figure 25.



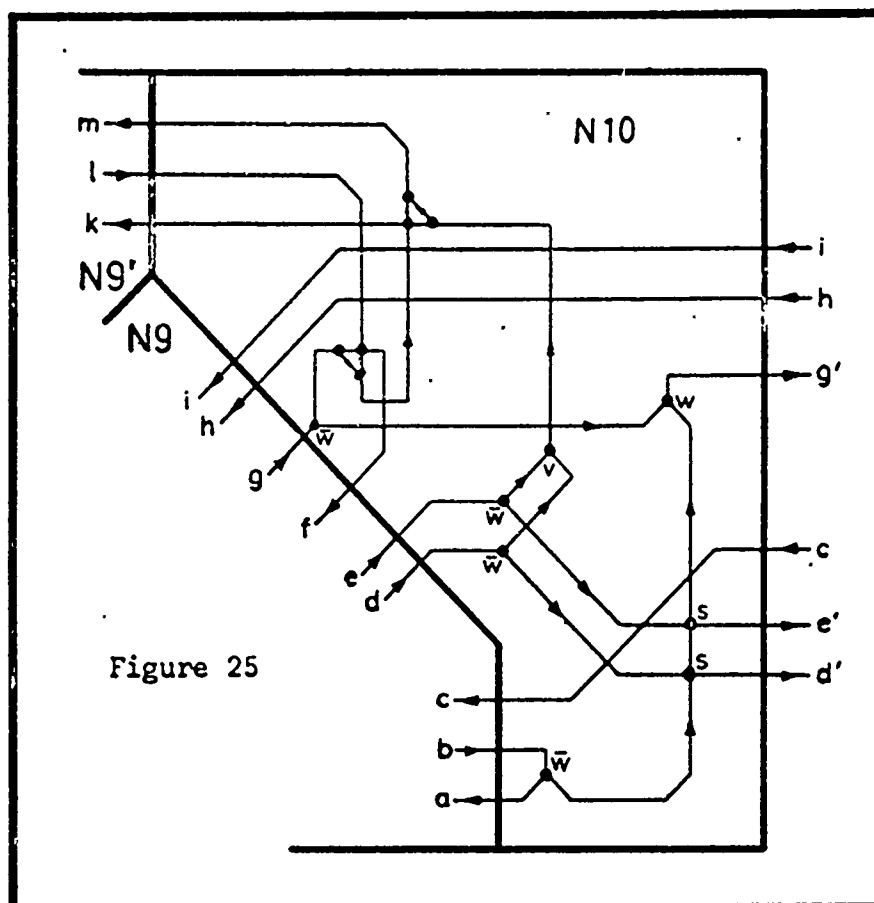


Figure 25

It thus appears that by means of such supplementation the  $\xi$ -operator can be eliminated by, proceeding from the periphery, intercoupling an N10-N10' pair between successive N9-N9' pairs (moving from one N9-N9' pair to the next). If we assume that the PD-store has a finite length, then this process will eventually terminate, and the memory will then consist of a chain of N11 nets (Figure 26), where the sections between the N11's will have the  $\xi$ -free C-form

$$C11: * \nabla / c \nabla d e / \nabla h i g .$$

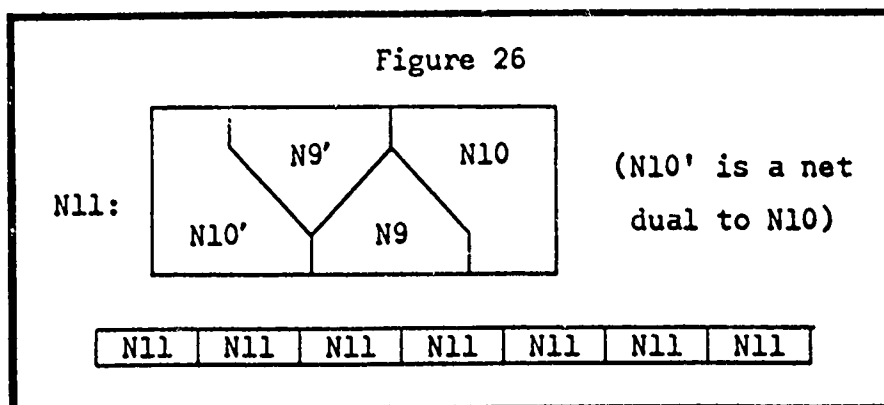


Figure 26

The fundamental difference between these two types of PD-automaton lies in the following: if an interrogation is directed to the periphery of an N11 chain meant for iterative use, then it is impossible for an answer to appear until an action chain has gone through all the copies of N11; further, the end of the memory must be occupied by a communication partner (net) with the C-form C11. If this communication partner has a plan that can be described by a finite net with the peripheral form C11, then this net must either be analyzable into separate nets, or contain conflicts, corresponding to the free choice (so to say) of the partner at the input of the chain, in the valuation of  $\nabla$ .

As an example we may cite the following plan of the end-partner: 'Received information (i. e. the information  $\nabla e d$  released from the PD-store) is lost; for any interrogation by the memory, the answer  $h$  is to be given (as in the left-shift of a computing register zeros are added to the right, and in a right-shift the information is lost).' Translated into the language of C-forms, this means that the valuation of the  $\nabla$ 's for the communication partner is the following:

$$* \nabla_1 / c \nabla_2 d e / \nabla_3 h i g .$$

$\nabla_1$  and  $\nabla_3$  are under the control of the input-partner; this is inherent in the sense of the memory utilization: one has the choice of either storing or interrogating, and if one chooses storing, then there is a choice in what is to be stored.  $\nabla_2$  brings about an accessing of information, and its valuation is not subject to the control of the input-partner.

In the net that is to represent the above plan of the end-partner the situation is reversed: only  $\nabla_2$  is under its control. The plan requires



that  $/c \nabla d e$  separate into  $/c d$  and  $/\emptyset e$  (i. e.  $\bar{Q}(e)$ ). Further,  $* \nabla \dots / \dots$  separates into  $* / \dots$  and  $* / \dots$ . Therefore, a signal cycle beginning, for example, with  $c$  at the input can be closed conflict-free only via  $/c d$  at the output, and an imagined 'infinite' N11-chain could not possibly work.

In the case of the PD-store consisting of only N9-N9' pairs the situation is different. For every activation at the input-periphery a 'wave' of actions propagates to the end, a wave that carries with it the combined information from  $\nabla_1$  and  $\nabla_3$ . It is only by means of the  $\&$ -operator (i. e. by the introduction of combined  $W-\bar{W}$ -cycles) that it is possible to make the desired feedback signals independent of the propagation of the wave and, therewith, of the length of the memory.

The only thing lacking, then, is the construction of the reading head of a Turing machine together with the tape field being observed; this is accomplished by the net N8 shown in Figure 27.

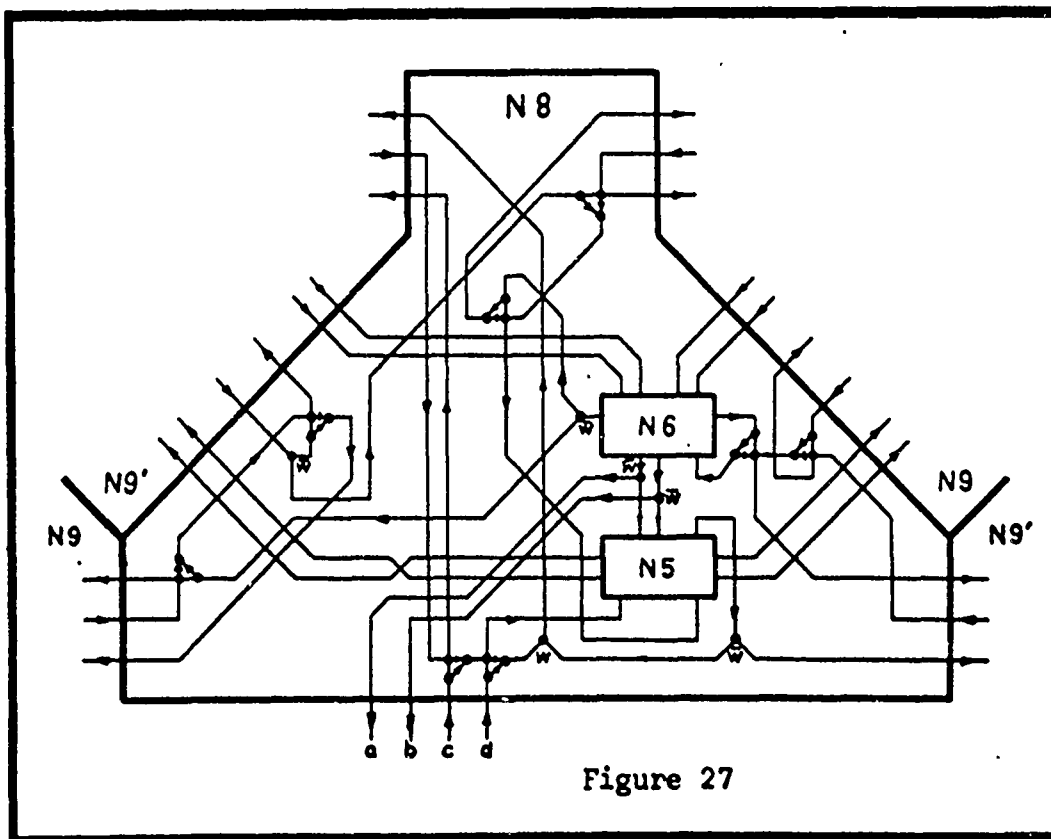


Figure 27

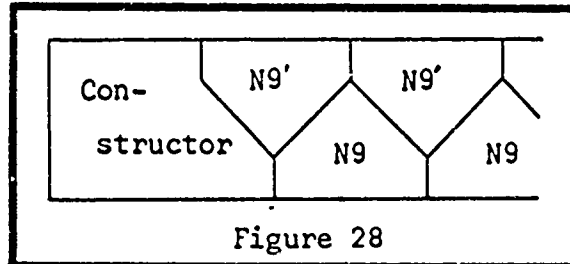
The construction of a net representation for the flow of information in a Turing machine is therewith concluded. There remain to be discussed the conditions on the ends of the tape, for there, too, the C-form C9 must be adhered to. However, since the tape may lose no information, and since nothing has been said about the coupling of the peripheral locations at the end of the tape, C9 is fulfilled trivially, that is by the absence of signals.

This necessity now gives us a prescription for the construction of such memories. This prescription is distinguished by the fact that it has a finite cycle and that a 'run' through this cycle is completely independent of the work cycle doing the constructing (here the tape). That means in this case that the *constructor* (see [2], p. 299) does not have to communicate with the user of the machine.

In the construction of a Turing machine one must therefore produce, in addition to a finite net, two identical N9-N9' constructors; the activation of the constructors simultaneously ends the possibility of communicating with them. Taking the situation of the lefthand half of the tape illustrated in Figure 28 as a beginning, the plan of a constructor will have to read as follows:

C1: \* / / / / / b a (N9) l m (N9) ,

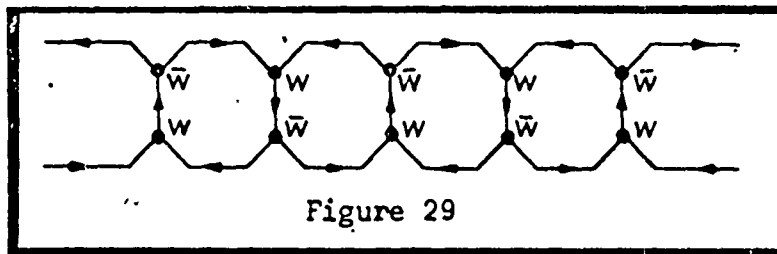
in which (N9) is a *constructive C-form* signifying the production of an N9 net. (In order to define this we should have to extend the range of the concepts used up to this point in order to describe the production of new locations and the process of connexion. Instead we shall mention only that the formal description is finite in all cases, and that the description would have little value without a statement of interpretation,



After a run through C1 the same situation arises; this fact is the justification of the  $\ast$ -operator. The essential parts of C1 are  $/ b a$  and  $/ \bar{l} m$ . Their interpretation clarifies the independence of the constructor. The nets N9 and N9' are constructed in such a way that they contain only one distinguished bit: the 'null' information in N6. This bit is not characteristic of the net, for after the determination of the net it can be shifted and replaced. On the other hand, by means of  $b$  a bit is introduced to the net N9 (or N9') that is always maintained in the net and enables it to be activated from the right. This characteristic distinguished bit (the *eigenbit*) is introduced by the event  $b$ , is confirmed by means of  $a$  for the constructor to the left, and is found at the location  $x$  between  $\bar{W}$  and  $W$  exactly as often as the C-form is run through  $n$  times ( $n \geq 0$ ). Its position at any given time corresponds to the observed position of the C-form. The course of the eigenbit through the C-form is analogous to the algorithm for the evaluation of CS2 formulas (v. p. 19).

The proof that the push-down automaton is capable of functioning conflict-freely, as well as insight into the *modus operandi* of the memory is to be had from an examination of the characteristic paths of the eigenbits (as shown in Figure 29). Figure 29 is obtained from the memory structure by the contraction of the  $/$ 's and the  $\nabla$ 's. There is always exactly one eigenbit in each cycle. If a cycle with no eigenbit were to

be added to the end, it would in no way influence the activity of the rest of the chain. Therefore one can undertake constructions on this cycle that correspond exactly to the productions of CS3, and as soon as the constructions are concluded the cycle may be given its eigenbit, after which time it can take part in the activities of the chain, without, however, disturbing the construction on the next cycle(s) (because its eigenbit can not leave it).



The topologization of the information flow in a Turing machine under consideration of the postulate P1 is therewith completed. In particular it has been shown that the C-form of the reading  $\ast / \nabla c d \nabla a b$  and therewith the C-form of the periphery of the universal-program matrix is independent of the activity of the constructors (and therewith of the length of the tape), so that the total net (that is to say, machine + constructors) contains no conflicts.

Therefore there exists no conflict-free experiment (algorithm) on the program periphery of the Turing machine that enables one to establish the length of the tape. In this sense, by virtue of the detachment of the constructors, the specified net is equivalent to an infinite net; that is to say, it is in the position to recognize irregular messages.

## 5. Formalization and Generalization

The construction of a net for a Turing machine has served us primarily in showing that the theory of synchronous automata is contained in a more general theory of communication forms. We must now speak of the applicability and further development of such a theory. As was mentioned in Section 1, the applications that were the impetus for this investigation are to be found in methods for the design and programming of information-processing machines.

Certain logical fundamentals of these applications have been provided in Section 4 by means of explicit patterning. We may now ask whether it is possible to interpret the formalism of the nets in some suitable way, and whether there exist material objects that have exactly the same behaviour as the intuitively-introduced switching elements of Section 4.

It is immediately clear that any of the conflict-free nets illustrated could be realized by imagining the nodes to be occupied by persons who have the task of acting in accordance with the actions of the particular node that they occupy. The N-structure of the net is then given by a spatio-topological structure of the communication possibilities between the various persons. It is shown therewith that conflict-free nets represent in all cases a workable form of organization with complete specification of the information flow.

In Section 4 we selected switching elements with functions that lie close to our understanding. However, to prove the general interpretability in the physical domain it would be necessary to express in the language of nets all of the primitive conceptual foundations of physics.

Such a task, however, would go far beyond the framework of this investigation; further it is seen to be far more to the point to test whether physical viewpoints should not be considered more even in the selection of the fundamental concepts and of the switching elements, before we even think of such a task as was mentioned above.

We should therefore like to investigate whether the concepts of Section 4 can not be derived from simpler and less numerous one; we undertake this investigation with the following threefold aim:

- 1) Postulate P2 should be able to be verified more easily (we have done this only for the W-nodes, and only sketchily for those);
- 2) the simplified concepts should be patterned after the physical concepts;
- 3) in principle it should be possible to give a complete description of any material body by means of a C-form.

It stands out immediately in our attempt at simplification that the node types U, V, and S are each subject to two actions. If it is possible to make do with 1-action nodes, then we can drop the concept of the node entirely and connect locations directly by means of actions. With definitions and axioms of the same form as in Section 4 plus the additional definition that each node be subject to exactly one action, with the elimination of the 'node' concept, we obtain, instead of the nets described above, a new sort of structure that we shall call an *action net* (or *A-net*).

In that we conceive of a node of a net as the superimposition of two actions, we recognize immediately that every net with the parameters

$P$  and  $M - 2$  is isomorphic to an A-net with the parameters  $P$  and  $M$ . Our interest, then, will be in A-nets with  $P = 4$  and  $M = 4$  (at most four locations participate in any given action; any given location takes part in at most four actions). The vicinity of two elements will mean participation in a common action; the vicinity of two actions will imply the existence of a common location. The valuation of a location with a bit will be changed in only one way, namely by the execution of an action.

In order to approach the goal cited we shall consider in what way a physical magnitude and the laws according to which it varies can be expressed by the distribution of bits in A-nets. We could effect this by imposing certain restrictions on the structure of a net, but it seems much more satisfactory to bring about a direct connexion between general physical principles (such as GP1 - GP3 of Section 3) and the properties of the actions. We wish, then, to admit only those actions that fulfill the general principles GP1 to GP3 according to any arbitrary interpretation thereof.

GP1, the existence of a quantized magnitude, we have taken into consideration even in the selection of the fundamental concepts. GP2, the validity of a law of conservation, we wish to fulfill by admitting only *conservative actions*: an action will be called *conservative* with respect to an object class  $i$  if the number of  $i$ -objects ( $i \in \{0, 1\}$ ) is the same in both the lefthand member of an action and the righthand member. For example, T is conservative with respect to the object classes, whereas W is not. GP3, the validity of a reaction principle, brings with the fact that any observation alters the observed property. The only way to exclude observation without alteration is to admit only *reactive ac-*

*tions*: an action will be called *reactive* if all of the objects named in its description are different (therefore if the 'state' of all participating locations is changed by the execution of the action). For example, the actions of Q, T, and W are reactive, those of V and S are not.

We do not wish to formalize the construction process here; we shall treat only those changes of object structures that consist in finite revaluations of locations by the execution of actions. In what follows, then, we shall consider only *nonsingular actions*: an action will be called *singular* if the set of locations named in the lefthand member of the action is different from the set of locations named in the righthand member.

A conservative, reactive, nonsingular action will be called a (*pure*) *transition*. In an action net composed entirely of transitions GP1 to GP3 are fulfilled for all physical magnitudes that can be expressed by sets of objects of a particular class in the interpretation. The totality of possible transitions can easily be enumerated:

01 → 10, 0011 → 1100, 000111 → 111000, ...

or in an easily understood 'orthography':

+-, ++-, +++--, ...

Every possible transition is contained in this sequence (no consideration is taken of the order of the locations).

It will be expected of an elementary process that it cover only a small number of locations. We now wish to ascertain, if it exists, the smallest number  $P$  such that the flow of effect in a corresponding net will have a nontrivial structure. For  $P = 2$  we have only the T-element of Section 4, and it is easy to convince oneself that using this element



alone only trivial structures can be produced.

On the other hand,  $P = 4$  may be seen to be sufficient. Indeed it may be seen that from the transition  $X: ++--$  alone, and using only locations with the same formal properties of symmetry, nets can be constructed for all automata and Turing machines. We can make do with locations with the following property: each location participates in exactly two actions that enable the revaluation  $0 \rightarrow 1$  in that location and in exactly two actions that enable the revaluation  $1 \rightarrow 0$  in that location. This has as a consequent that we can imagine every event at a location as connected with the actual flow of information (transport of a bit): in the valuation of a location there are exactly two actions  $X_1$  and  $X_2$  that can introduce an event, and the action will actually enter by means of exactly one of the actions, so that the decision content of the bits named is expressed by  $\vee X_1 X_2$ . We can therefore formally assign to each location a C-form (in CS3):

$$C_S: * / \vee X_1 X_2 \vee X_3 X_4 ,$$

and, similarly, to each X-transition we can assign a C-form:

$$C_X: * / \& a b \& c d .$$

It is only with this formal relation that we are provided with a connexion with the usual concept of information, and this relation makes the discovery of the following nets much easier.

Let us now examine once again the concept of conflict. In a conflict-free net every decision is made solely on the basis of a situation expressed in the net; no bit can, so to say, be born. We can imagine, then, that the bit that decides a two-action conflict must be introduced to the net from outside. If we assume that this can take place only by

means of actions, then we must view a non-conflict-free net as *incomplete*. Since a net that is capable of communication is by its very nature incomplete, we shall be interested primarily in incomplete nets anyhow. We can therefore view a conflict-free net as a net with a completely specified information supply.

For the abstract description of a physical system the sense of temporal direction is quite without consequence as long as we preserve its assignment to the directional sense of the flow of quantities concerned. If we reverse the sense of direction ' $\rightarrow$ ' of the actions of a net, then we effect an interchange of the influx and the efflux of information.

The reversibility of the ' $\rightarrow$ ' can not be just imagined away for physical elementary processes. Therefore we wish to express with this symbol only an abstract sense of orientation that makes an assignment of temporal sense to the directional sense of the flow. If we reverse the ' $\rightarrow$ ' in a conflict-free net, the net does not in general remain conflict-free. The conflicts that arise indicate exactly those locations to which we must supply information, thus those from which we derived information before the reversal of the ' $\rightarrow$ ' (for example, the V-element, or the or-gate).



We shall call an action net *versible* if it is conflict-free for both valuations of the ' $\rightarrow$ '. In a verisible net, therefore, information can be neither produced nor lost, and all of the locations of influx or efflux are well-defined. We shall not pursue the relation between verisible A-nets and *reversible* physical processes any further at this point.

Such simple action nets are possibly of value for the discussion of our conceptions of models of the fine structure of matter, for if we

admit postulate P1 as a generally valid physical principle -- and for the investigation of fundamentals this seems to be quite advantageous -- then we obtain with a theory of action nets (or of information flow) a useful mathematical tool for the theoretical-physical description of those magnitudes that conform to the requirements mentioned in postulate P2. This is especially useful because it will render superfluous the embedding of the quantities to be observed in a continuum the properties of which can be selected quite arbitrarily when they go beyond what is observable, indeed properties that represent an unnecessary burden in the realm of fundamental questions.

We now wish to convert the most important nets of Section 4 into versible A-nets; in this conversion we shall use only the transition X and locations of the type mentioned above. Since the nets can not be complete it will be impossible for all of the actions and locations to fulfill the above definitions; we therefore change the definitions by replacing 'exactly two' by 'at most two'. The *periphery* of the action net comprises all the locations and actions that are incompletely connected in the above sense.

Since  $M = 4$  we can no longer use the graphs of Section 4 for the pictorial representation. We therefore make the following conventions:

-  location with 0-bit at the moment of construction
-  location with 1-bit at the moment of construction

Action  $X_i$  (a b c d) : + + - -

The pictorial representation of the two possible (arbitrary) choices of action  $X_i$  is given in Figure 30.

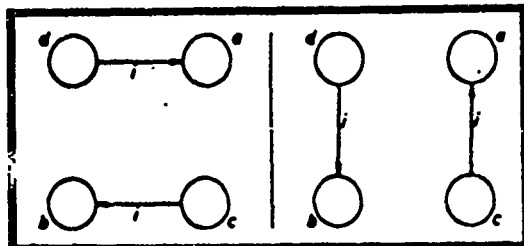
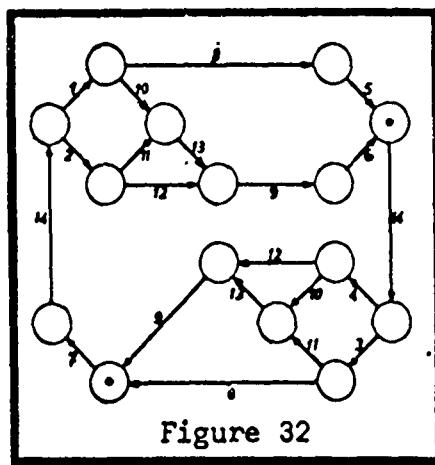
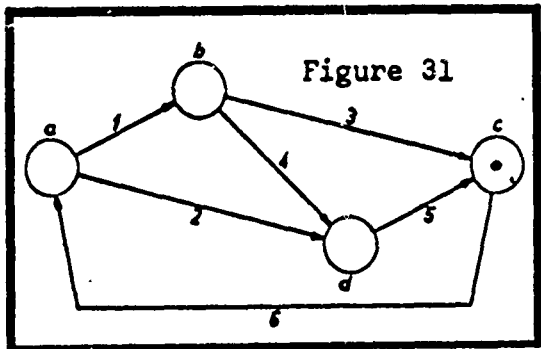


Figure 30

An action is represented by a pair of directed line segments bearing the same natural number as a marker. Different actions will have different markers.

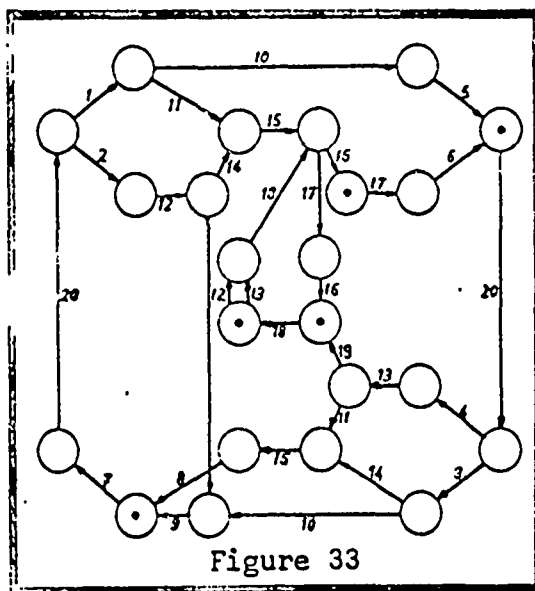
Predicate calculus functions of two variables: It is sufficient to give the representation of the eight unanalyzable '1:3' functions since the Sheffer function is included among them. With disjunction we should have the following (with reference to Figure 31): 6 is activation;  $\nabla 1 2 = x$ ;  $\nabla 3 4 = y$ ;  $\nabla 3 5 = (x \vee y)$ ;  $\nabla 4 2$  is the information released. In the case of conjunction we should have the following:  $\nabla 2 1 = x$ ;  $\nabla 4 3 = y$ ; and  $\nabla 5 3 = (x \wedge y)$ ; etc.



If it is always the case that both x and y must be interrogated, as in the 'disjunction' example of Section 4, then some complete transitions will appear; in Figure 32, then, we should have the following: 7 is the activation;  $\nabla 1 2 = x$ ;  $\nabla 3 4 = y$ ; and  $\nabla 5 6 = (x \vee y)$ .

This arrangement is of advantage to us in a non-conflict-free net only if the decisions x and y require a 'greater time span' if they are

activated serially than if they are activated simultaneously. (The concept of time span is meaningful only in non-conflict-free nets.) Then, however, we can take the conflict into the evaluation of the function (as in Section 4) and make the greatest possible use of the time saved. Further, here it is easy (in contrast to Section 4) to design the net for arbitrarily frequent use. Thus, in Figure 33, we have the following: 7 is the activation;  $\nabla 1 2 = x$ ;  $\nabla 3 4 = y$ ;  $\nabla 5 6 = (x \vee y)$  (accelerated);  $\nabla 12 13$  is conflict with  $x = y = 1$ ; and  $\nabla 8 9 = (x \equiv y)$ . The net, as is noted, also gives us the equivalence function, which, of course, cannot be accelerated.



Functions of Automata. As in Section 4 the alphabets for input and output will have two elements each,  $a$  and  $b$ , and  $c$  and  $d$ , respectively. The normal C-form is  $* / \nabla a b \nabla c d$ . We now wish to place value on the *reversibility* of all automata (i. e. exchangeability of input and output), which appears only in exceptional cases in the usual theory. In the following net representation it arises quite of itself that all information lost to the automaton must be released at a well-defined location. The communication via this *latent peripheral location* thus sup-

plements the information flow in such a way that there is also a *law of the conservation of information*. This is generally the case for versible action nets.

Certain questions of the minimization of automata can, on the basis of these considerations, be answered surprisingly simply, as will be shown in another place. Here we wish to leave such questions aside. In this sense the generalization to automata with arbitrary finite alphabets should be thought of as carried out in such a way that a recoding (as in Sections 2 and 4) into and out of  $\{0, 1\}$  takes place by means of two especially simple automata, and so that the state-graph of the 'central' automaton is correspondingly modified.

If  $Q = \{q_1, \dots, q_n\}$  be a set of states,  $B = \{a, b\}$  and  $C = \{c, d\}$  the alphabets for input and output, then every automaton is represented by a single-valued mapping  $M$  of  $Q \times B$  into  $Q \times C$ . There always exist alphabets  $B' = \{e, f\}$  and  $C' = \{g, h\}$ , a set  $R = \{r_1, \dots, r_m\}$ , and a one-to-one mapping  $F$  of  $(Q \times B) \cup (R \times B')$  onto  $(Q \times C) \cup (R \times C')$  such that if  $F'(C') = e$  then  $(FF')^i F = M$ , where  $0 \leq i \leq m \leq 2n$ . The following may be taken as an example:

$$\begin{array}{l}
 M: \begin{array}{l} q_1 a \ c \ q_2 \\ q_1 b \ c \ q_2 \\ q_2 a \ c \ q_1 \\ q_2 b \ d \ q_1 \end{array} \\
 F: \begin{array}{l} q_1 a \ g \ r_1 \\ q_1 b \ h \ r_1 \\ q_2 a \ c \ q_1 \\ q_2 b \ d \ q_1 \\ r_1 e \ c \ q_2 \\ (r_1 f \ d \ q_2) \end{array}
 \end{array}$$

The efflux of information is thus centered on  $F'$ , and we have introduced finitely many intermediate states  $r$ . The structure of the automaton is expressed by a permutation of  $(Q \cup R) \times \{0, 1\}$ , and conversely

every such permutation represents an automaton.

Now we have only to describe the net N7 and to identify the free peripheral locations  $a'$ ,  $b'$ ,  $c'$ , and  $d'$  in accordance with the permutation cited; this part of the coupling prescription, the part that produces a special automaton from an N7-chain, is already taken care of. The direct translation of N7 into the language of this Section leads to a non-versible net of 36 transitions. By observing the information flow it may be seen at which locations a superfluous bit-transport takes place, and by the omission of these locations the net shown in Figure 34 results, a net with only four interior locations  $x$ ,  $y$ ,  $z$ , and  $u$ . By reflecting N7 about a horizontal we obtain N7". The total net has the form shown in Figure 35.

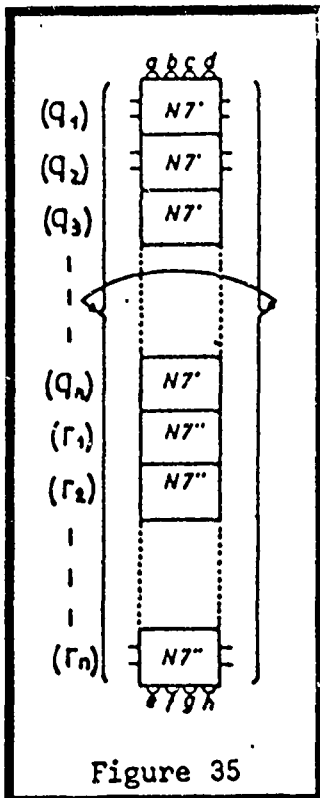


Figure 35

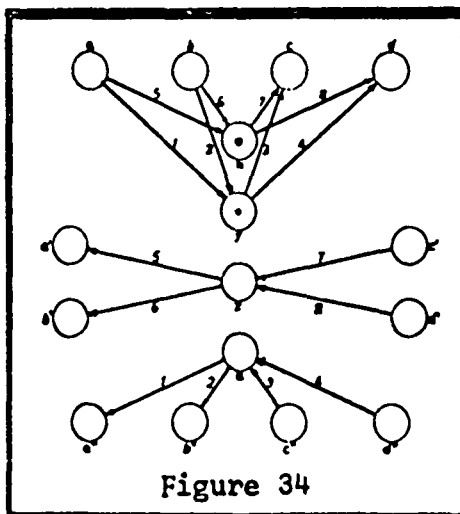


Figure 34

In the uppermost net we now interchange the valuations of  $x$  and  $z$  (at construction); the automaton is then in state  $q_1$  and is ready to function. The automaton will work as long as the utilized parts of the automaton function  $M$  form a one-to-one

partial function of  $M$ ; it then attempts to give off a bit  $\nabla g h$ , and we must take precautions that the C-form  $\ast / \nabla g h e$  is fulfilled at the latent periphery, either by means of a non-versible net (as in Figure 36)

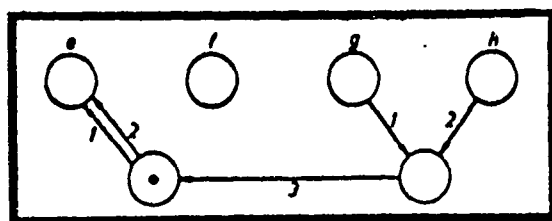


Figure 36

or by means of a versible push-down store. Such a store is given, however, by the net pair N9-N9' by concatenation, and the existence of a versible A-net with this function proves at the same time the representability of the two halves of the tape of a Turing machine. In order to show the type of concatenation we specify two net pairs. Each net pair carries two bits of the stored information. For the righthand half-tape ... E F E F E F ... we show the concatenation E F in Figure 37, that of F E in Figure 38.

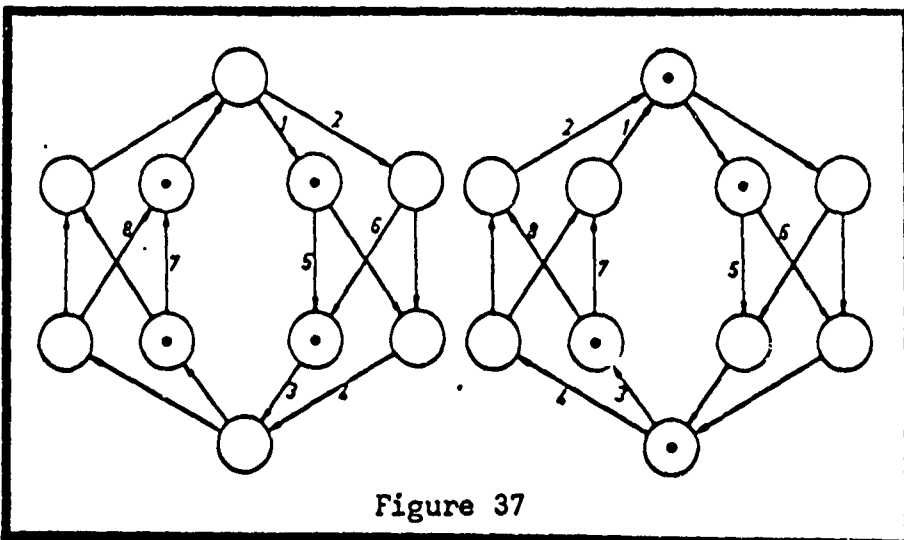


Figure 37

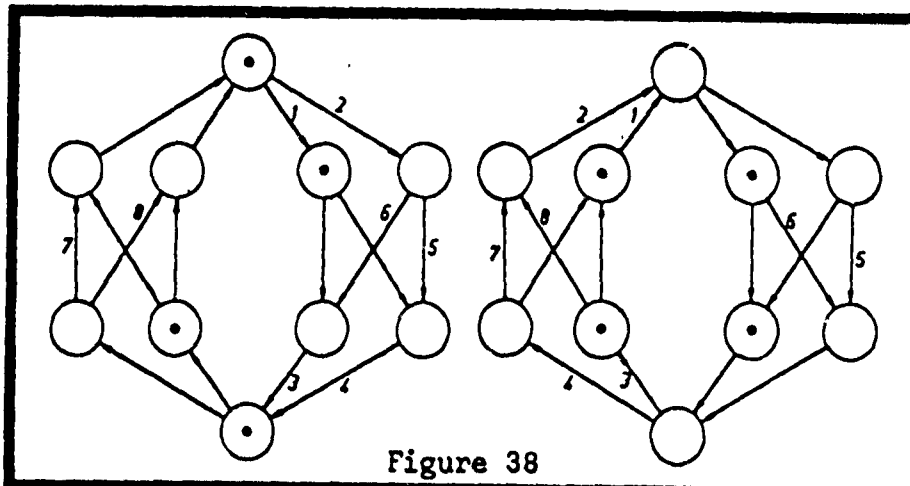


Figure 38



The question of the technical realization of the switching elements considered is not to be discussed here for the following reasons: 1) The question is not relevant in this form. The selection of the switching elements was purely arbitrary (cf. Section 4). The only essential thing is adherence to the communication forms on the periphery of arbitrarily composed, connected N-structures. 2) The fact that there is no longer any forced synchronization of the actions facilitates the technical interpretation. The question does arise as to how one can secure the invariance of the actions. In the macroscopic-mechanical realm the question is certainly satisfactorily soluble (for example in switching theory, in the inscription, movement, and reading of material data carriers, and in communication between persons by means of speech, writing, or transmitted objects); and so also in the macromolecular realm obviously (for example, in the nervous system) and possibly with complete rigor in the atomic domain. In the field of electronics the question has not been investigated -- probably because one is accustomed to observing asynchronous processes in a continuum (at least for time) and because on occasions for discontinuous observation it is maintained that the partitioning of time into a completely ordered set of individual steps is always a meaningful process that in no way limits generality for all practical purposes. We must certainly add that this is erroneous in consideration of Sections 2 to 4. It is therefore necessary to think through the question of the immediate technical representation of action nets in such a way that we entirely avoid the detour via the usual time concept for automata.

3) The assumption would seem to be justified that the selection of switching elements in Section 5 on the basis of physical considerations

in no way complicates the realization; indeed with suitable interpretation it accommodates the realization. Further discussion of these questions must be left to the specialists, however.

We assemble once again the advantages expected of this manner of looking at things: 1) the possibility of conflict-free communication with and between automata; 2) the independence of the construction procedures of the work processes of automata, and therewith the unlimited possibility of extending automata during their work; 3) the simplification of the interpretation in consideration of the fact that switching elements are required only to have 'the shortest possible switching time' without being required to adhere to tolerances in the switching time by the logical structure; it should be noted in that context that the concept of switching time can appear only in those total nets that have at least one location of conflict; 4) the conflict-free coupling of processes running simultaneously; 5) the displacement of noneliminable conflicts to locations where we can derive benefit from their existence (as in the or-gate).

We return now to the previously-mentioned (Section 1) application to questions of programming. This possibility of application is so obvious that we need only allude to the conceptual similarity of the systems that have been illustrated to the conventional flow chart. Once again it is the unconventional treatment of temporal dependencies that allows us a completely explicit representation of the flow of effect and therefore a rigorously formal treatment of communication with consideration of the physical possibilities. The important rôle of the parallelity operator has already been discussed in Section 4. Its intro-

duction can be explained as the renunciation and replacement of a universal (and, therefore, usually unmentioned) parallelity operator that is obligatory for each step in time and connects *all* processes in two successive time steps; its replacement is effected by a uniformly limited connexion that is to be specified explicitly. It is clear that this has a far-reaching influence on the description of organizational compositions.

Without such an operator no meaningful interpretable formalization of irregular communication processes is possible. Even beyond this the concept of conflict allows a formally exact, finite representation of such expressions as 'NA is trying to solve the problem ', and this representation is independent of any statement of the content or the structure of the problem B. The conflict concept further yields a representation of the *game* and allows a distinct separation of the various concepts of *probability*; this separation is accomplished by means of the explicit specification of nets that fulfill the prescriptions (C-forms) for communication for the purpose of 'ascertaining a probability'. By this we mean a plan for the derivation of decisions  $\forall x y$  from a probability judgement; such judgements are to be had as output from nets whose freedom from conflict is not proven.

In this way certain very simple principles of organization invade the domain of activity designated here as programming, concepts that are beyond the reach of ordinary 'time-step programming' as a matter of principle. This is of interest because of the growing complexity of the tasks that are to be given to information-processing machines; with irregular tasks there is the danger that the user of the machine will no

longer be capable of overseeing the set of conventions that must be adhered to. Since all irregular parts of the problems must, however, be mapped onto such conventions, even if an automaton were to take over a partial plan of the user this would represent only a comparatively trivial lightening of the user's job if the partial plan is expressible in time-step programming.

In this sense, then, the problem of automation (or automaticization) ought to be viewed to a greater degree as a problem of communication. The mathematico-logical treatment of this problem can, so it is to be hoped, be eased by the development of an interpreted calculus of C-forms. The nets illustrated in this work were found in an approach to this development; further, such approaches have also yielded valuable suggestions for the translation of formal languages such as ALGOL. Therefore, we should like to return to this calculus once again.

In a way analogous to the transition from CS1 to CS2 we generalize CS3 to CS4 with the two-place operators  $\forall$ ,  $/$ , and  $\&$  and the zero-place operator  $R$ . For convenience we also admit operators derivable from these, such as  $\ast$  and a cycle-operator  $Z$ :

$$\ast a \equiv \forall a / a R; Z a b \equiv \forall a / / a b R .$$

An  $R$  in the second member of a  $\forall$ -formula leads, as in CS2, to a recursion. By means of  $R$  we can express the activation of a constructor. The interpretation of  $R$  has not yet been investigated sufficiently. It is surmised that with the aid of CS4 formulas and a suitably defined and interpreted  $\equiv$ -relation between them all of the elementary aspects of communication can be satisfactorily represented. The conjecture could be supported quite strongly by a pair of theorems analogous to T1 and T2.

A corresponding formal proof is possible for certain action nets, but because of certain difficulties of interpretation that have not yet been put to rest this proof will not be reproduced here.

In other words, there is yet lacking a sufficient characterization of the class of generalized automata for which a pair of theorems like T1 and T2 (but with CS4 instead of CS1) is valid. On the other hand, the author hopes, with this work, to have contributed to the clarification of the conceptual foundations of a theory of communication.

o o o

Selected Bibliography

- [1] Bottenbruch, H., *Uebersetzung algorithmischer Formelsprachen in die Programmsprache für Turingmaschinen*. Dissertation. Darmstadt, 1957.
- [2] Burks, A. W., Computation, behavior and structure in fixed and growing automata, in *Self-Organizing Systems*, Papers of the Interdisciplinary Conference on Self-Organizing Systems, Chicago, 1959 (M. C. Yovits, ed.), pp. 282-311. New York, 1960.
- [3] Burks, A. W. and Wang, H., The logic of automata. *Journal of the Association for Computing Machinery* 4, 193-218 and 279-297 (1957).
- [4] Burks, A. W. and Wright, J. B., Theory of logical nets. *Proceedings IRE* 41, 1357-1365 (1953); reprinted in Moore [17] pp. 193-212.
- [5] Carnap, R., *Logical Foundations of Probability*, Chicago, 1950.
- [6] Carnap, R., *Introduction to Symbolic Logic and Its Applications* (trans. by W. H. Meyer and J. Wilkinson). New York, 1958.
- [7] Chomsky, N., Formal properties of grammars, in *Handbook of Mathematical Psychology, Volume II* (Luce, Bush, and Galanter, eds.), pp. 323-418. New York, 1963.
- [8] Copi, I. M., Elgot, C., and Wright, J. B., Realization of events by logical nets, *Journal of the Association for Computing Machinery* 5, 181-196; reprinted in Moore [17] pp. 175-192. (1958).
- [9] Curry, H. B. and Feys, R., *Combinatory Logic*. Amsterdam, 1958.
- [10] Davis, M. D., *Computability and Unsolvability*. New York, 1958.
- [11] Kleene, S. C., Representation of events in nerve nets and finite automata, in *Automata Studies* (C. E. Shannon and J. McCarthy, eds.), pp. 3-42. Princeton, 1956.
- [12] Lee, C. Y., Automata and finite automata, *Bell System Technical Journal* 39, 1267-1295 (1960).
- [13] McCarthy, J., Recursive functions of symbolic expressions and their computation by machine, *Communications of the Association for Computing Machinery* 3, 184-195 (1960).
- [14] McNaughton, R. and Yamada, H., Regular expressions and state graphs for automata, *IRE Transactions on Electronic Computers* EC-9,

- 39-57 (1960); reprinted in Moore [17] pp. 157-174.
- [15] Mealy, G. H., A method for synthesizing sequential circuits, *Bell System Technical Journal* 34, 1045-1079 (1955); reprinted in Moore [17] pp. 157-174.
- [16] Moore, E. F., Gedanken experiments on sequential machines, in *Automata Studies* (C. E. Shannon and J. McCarthy, eds.), pp. 129-153. Princeton, 1956.
- [17] Moore, E. F., *Sequential Machines: Selected Papers*. Reading, Mass., 1964.
- [18] Muller, D. E. and Bartky, W. S., A theory of asynchronous circuits, in Proceedings of an International Symposium on the Theory of Switching, Harvard University, April 1957, *Annals No. 29 of the Computation Laboratory of Harvard University*. Cambridge, Mass., 1959.
- [19] Post, E. L., Finite combinatory processes -- Formulation I, *Journal of Symbolic Logic* 1, 103-105 (1936).
- [20] Rapoport, A., *Operational Philosophy*. New York, 1953.
- [21] *IBM Journal of Research and Development* 4, 208ff. (1960).

Security Classification

DOCUMENT CONTROL DATA - R&D		
<i>(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)</i>		
1 ORIGINATING ACTIVITY (Corporate author)  Dr. Carl Adam Petri Bonn University, Germany	2a. REPORT SECURITY CLASSIFICATION  Unclassified	2b. GROUP
3 REPORT TITLE  Communication with Automata		
4 DESCRIPTIVE NOTES (Type of report and inclusive dates)  Supplement I to RADC-TR-65-377, Volume I		
5 AUTHOR(S) (Last name, first name, initial)  Dr. Carl Adam Petri		
6 REPORT DATE  January 1966	7a. TOTAL NO OF PAGES  89	7b. NO. OF REFS  21
8a. CONTRACT OR GRANT NO. 30(602)-3324  b. PROJECT NO.  c.  d.	8a. ORIGINATOR'S REPORT NUMBER(S)  8b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) RADC-TR-65-377, Vol. I (Final Report)	
10 AVAILABILITY/LIMITATION NOTICES  Distribution of this document is unlimited.		
11 SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY  RADC, GAFB. NY 13440	
13 ABSTRACT The theory of automata is shown not capable of representing the actual physical flow of information in the solution of a recursive problem. All of the really useful results of automata theory may be expressed by means of these new concepts. Moreover, the results retain their usefulness and the new procedure has definite advantages over the older ones. The proposed representation differs from each of the presently known theories concerning information on at least one of the following essential points: 1. The existence of a metric is assumed for neither space nor time nor for other physical magnitudes. 2. Time is introduced as a strictly local relation between states. 3. The objects of the theory are discrete, and they are combined and produced only by means of strictly finite techniques. The following conclusions drawn from the results of this work may be cited as of some practical interest: 1. The tolerance requirements for the response characteristics of computer components can be substantially weakened if the computer is suitably structured. 2. It is possible to design computers structurally in such a way that they are asynchronous, all parts operating in parallel, and can be extended arbitrarily without interrupting their computation. 3. For complicated organizational processes of any given sort the theory yields a means of representation that with equal rigor and simplicity accomplishes more than the theory of synchronous automata.		

DD FORM 1473  
1 JAN 64

Security Classification



14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
	Computers Theory Data Processing Systems					

**INSTRUCTIONS**

1. **ORIGINATING ACTIVITY** Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization (*corporate author*) issuing the report.
- 2a. **REPORT SECURITY CLASSIFICATION:** Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.
- 2b. **GROUP:** Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.
3. **REPORT TITLE:** Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.
4. **DESCRIPTIVE NOTES:** If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.
5. **AUTHOR(S):** Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.
6. **REPORT DATE:** Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.
- 7a. **TOTAL NUMBER OF PAGES:** The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.
- 7b. **NUMBER OF REFERENCES:** Enter the total number of references cited in the report.
- 8a. **CONTRACT OR GRANT NUMBER:** If appropriate, enter the applicable number of the contract or grant under which the report was written.
- 8b, 8c, & 8d. **PROJECT NUMBER:** Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.
- 9a. **ORIGINATOR'S REPORT NUMBER(S):** Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.
- 9b. **OTHER REPORT NUMBER(S):** If the report has been assigned any other report numbers (*either by the originator or by the sponsor*), also enter this number(s).
10. **AVAILABILITY/LIMITATION NOTICES:** Enter any limitations on further dissemination of the report, other than those

imposed by security classification, using standard statements such as:

- (1) "Qualified requesters may obtain copies of this report from DDC."
- (2) "Foreign announcement and dissemination of this report by DDC is not authorized."
- (3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through \_\_\_\_\_."
- (4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through \_\_\_\_\_."
- (5) "All distribution of this report is controlled. Qualified DDC users shall request through \_\_\_\_\_."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. **SUPPLEMENTARY NOTES:** Use for additional explanatory notes.
12. **SPONSORING MILITARY ACTIVITY:** Enter the name of the departmental project office or laboratory sponsoring (*paying for*) the research and development. Include address.
13. **ABSTRACT:** Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as (TS), (S), (C), or (U)

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. **KEY WORDS:** Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.