Theoretical Computer Science 10 (1980) 19–35
© North-Holland Publishing Company

ON EQUATIONS FOR REGULAR LANGUAGES, FINITE AUTOMATA, AND SEQUENTIAL NETWORKS*

J.A. BRZOZOWSKI
E. LEISS**
Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

Communicated by M. Nivat
Received November 1977
Revised November 1978

Abstract. We consider systems of equations of the form

$$X_i = \bigcup_{a\in A} a \cdot F_{i,a} \cup \delta_i, \quad i = 1, \ldots, n$$

where A is the underlying alphabet, the X_i are variables, the $F_{i,a}$ are boolean functions in the variables X_i, and each δ_i is either the empty word or the empty set. The symbols $\cdot$ and $\cup$ denote concatenation and union of languages over A. We show that any such system has a unique solution which, moreover, is regular. These equations correspond to a type of automaton, called boolean automaton, which is a generalization of a nondeterministic automaton. The equations are then used to determine the language accepted by a sequential network; they are obtainable directly from the network.

1. Notation

Let A be a finite alphabet and A^* the free monoid generated by A. An element of A is called a letter and an element of A^* is called a word over A. The unit element of the monoid A^* is the empty word λ. The length $|w|$ of a word w over A is the number of letters in w; note that $|\lambda| = 0$. The concatenation (product in the monoid A^*) of two words u and v is denoted by $u \cdot v$. The reverse w^ρ of a word w over A is defined recursively: $\lambda^\rho = \lambda$, and $(va)^\rho = av^\rho$ for $a \in A$, $v \in A^*$.

A subset of A^* is called a language over A. The empty language is denoted by $\emptyset$, and I is a shorthand for the language A^*. The concatenation of two languages L and L' is $L \cdot L' = \{u \cdot v \mid u \in L, v \in L'\}$. If L is a language, then $L^* = \bigcup_{n\geq 0} L^n$, where $L^0 = \{\lambda\}$. The left quotient $w\backslash L$ of a language L over A with respect to a word w over A is the language $\{x \mid wx \in L\}$; similarly for the right quotient, $L/w = \{x \mid xw \in L\}$. The reverse L^ρ of a language L is the language $\{w^\rho \mid w \in L\}$.

* This research was supported by the Natural Sciences and Engineering Research Council, Canada under grant No. A-1617.
** Present address: Department of Computer Science, University of Kentucky, Lexington, KY 40506, U.S.A.

The set $P(A^*)$ of all languages over A together with the set operations union ($\cup$), intersection ($\cap$), and complement ($^-$) forms a boolean algebra, in which $\emptyset$ and I act as zero and one, respectively.

We also consider the finite boolean algebra L_n of 'language' functions $f: \mathsf{X}_{i=1}^{n} P(A^*) \to P(A^*)$, i.e. the functions which can be expressed in terms of unions, intersections, and complements of the variables. (Note that $\mathsf{X}_{i=1}^{n} S$ denotes the cartesian product of n copies of S.) The constant functions $\emptyset$ and I act as zero and one, respectively.

Another finite boolean algebra which will be used is the set B_x of boolean functions $f: \mathsf{X}_{i=1}^{n}\{0,1\} \to \{0,1\}$ in the variables $x_1, \ldots, x_n$, x being $\{x_1, \ldots, x_n\}$, together with the operations OR ($\vee$), AND ($\wedge$), and complement ($'$). The constant functions 0 and 1 act as zero and one, respectively.

Clearly, B_n and L_n are isomorphic as boolean algebras. The following correspondence will be used:

	B_n	L_n
constant functions:	0	$\emptyset$
	1	$I = A^*$
variables:	$y = y_1, \ldots, y_n$	$X = X_1, \ldots, X_n$
operators:	$\wedge$	$\cap$
	$\vee$	$\cup$
	$'$	$^-$
function symbols:	$f_{i,j}(y)$	$F_{i,j}(X)$
	$g(y)$	$G(X)$

We obtain $F_{i,j}$ from $f_{i,j}$ as follows. Take any expression for $f_{i,j}$ (such as the canonical sum of products) involving $0, 1, y_1, \ldots, y_n, \wedge, \vee$ and $'$. In this expression replace y_i by X_i, $i = 1, \ldots, n$, 0 by $\emptyset$, 1 by I, $\wedge$ by $\cap$, $\vee$ by $\cup$, and $'$ by $^-$. We now have an expression in the variables $X_1, \ldots, X_n$. The language function defined by this expression is precisely $F_{i,j}(X_1, \ldots, X_n)$. The function G is obtained from g in the same way.

We briefly review the concept of finite automaton. A (nondeterministic) finite automaton $\mathscr{A}$ is a quintuple

$$\mathscr{A} = (A, Q, \tau, Q_0, F)$$

where A is the input alphabet, Q is the finite (nonempty) set of states, $Q_0 \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, and $\tau: Q \times A \to P(Q)$ is the transition function, where $P(Q)$ denotes the power set of Q. If Q_0 and $\tau(q, a)$ for all $q \in Q$ and $a \in A$ contain exactly one element, $\mathscr{A}$ is called deterministic. The function τ is extended to $P(Q) \times A^*$ in the usual way. We assume that $\mathscr{A}$ is connected i.e. for any $q \in Q$ there exists some $w \in A^*$ such that $q \in \tau(Q_0, w)$. A word $w \in A^*$ is accepted by $\mathscr{A}$ iff $\tau(Q_0, w) \cap F \neq \emptyset$. $L(\mathscr{A})$, the set of words accepted by $\mathscr{A}$, is a regular language, and to each regular language R corresponds a unique deterministic automaton $\mathcal{A}_0$ with the minimal number of states such that $\mathcal{A}_0$ accepts R; $\mathcal{A}_0$ is called the reduced automaton of R. The reverse $\mathcal{A}^\rho$ of a deterministic finite automaton $\mathcal{A} = (A, Q, \tau, q_0, F)$ is defined as follows: Let $Q_w = \{q \in Q \mid \tau(q, w) \in F\}$. Then $\mathcal{A}^\rho = (A, P, \mu, p_0, G)$, where

$$P = \{p \mid p = Q_w \text{ for some } w \in A^*\},$$

$$p_0 = F,$$

$$G = \{p \in P \mid q_0 \in p\}$$

and

$$\mu(p, a) = \{q \in Q \mid \tau(q, a) \in p\} \quad \text{for } p \in P \text{ and } a \in A.$$

$\mathcal{A}^\rho$ is always reduced and the language accepted by $\mathcal{A}^\rho$ is precisely the reverse of the language accepted by $\mathcal{A}$, $L(\mathcal{A}^\rho) = [L(\mathcal{A})]^\rho$ (see [3]).

2. Introduction

Consider the following set of language equations over the alphabet $\{a, b\}$

$$X_1 = a \cdot X_1 \cup b \cdot X_2 \cup \lambda,$$

$$X_2 = a \cdot X_2 \cup b \cdot X_1,$$

to be solved for X_1.

It is well known that these equations correspond to a deterministic finite automaton, namely the automaton given by Fig. 1 where an arrow ($\rightarrow$) indicates an initial state, and double circles denote a final state.

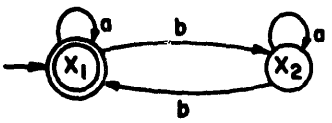

Fig. 1.

Conversely, given any deterministic automaton we easily derive a set of equations as above. Both concepts define the same (regular) language.

The important observation for us is that on the right-hand side of the equations only single variables appear.

There is a well known generalization of this notion which allows not only single variables but unions of variables on the right-hand side of the equations. An example is

$$X_1 = a \cdot X_2 \cup b \cdot \emptyset \cup \lambda,$$
$$X_2 = a \cdot (X_1 \cup X_2) \cup b \cdot X_1,$$

to be solved for $X_1 \cup X_2$.

This set of equations corresponds to the nondeterministic automaton repıesented by Fig. 2.

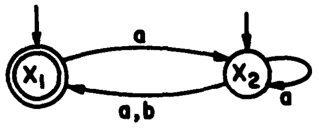

Fig. 2.

It seems natural to introduce a further generalization, that is rather than allowing single variables or unions of variables we will allow boolean functions of variables. For example, consider the following equations.

$$X_1 = a \cdot \bar{X}_2 \cup b \cdot I,$$
$$X_2 = a \cdot X_2 \cup b \cdot (X_1 \cap \bar{X}_2) \cup \lambda,$$

to be solved for $X_1 \cap X_2$.

In this paper we will study systems of such equations. In particular we will show that there is always a unique solution which, moreover, is regular (Section 3). We also define the automata, called boolean automata, corresponding to these systems of equations, and indicate their relation to deterministic automata (Section 4).

A completely different, but no less important motivation is provided by the fact that these systems of equations can be used to relate sequential networks to regular languages in a very direct way (Section 5). (A method of obtaining the regular language accepted by a given sequential network has been described in [4]; however the basic approach used there was different and less direct.)

3. Left language equations

In this section we will first define formally the equations we propose to study, and then prove that they always have a unique solution; this solution is regular.

A system of *left language equations* has the following form:

$$\begin{aligned} X_i &= \bigcup_{j=1}^{m} a_j \cdot F_{i,j}(X) \cup \delta_i, \quad i = 1, \ldots, n \\ X_0 &= G(X) \end{aligned} \qquad (1)$$

where $X = X_1, \ldots, X_n$, δ_n, $\delta_i \in \{\lambda, \emptyset\}$, $F_{i,j}$ is a language function in L_n for $i = 1, \ldots, n$ and for $j = 1, \ldots, m$, and G is also in L_n; let $A = \{a_1, \ldots, a_m\}$. The term 'left language equation' reflects the fact that the letters a_j of the alphabet appear on the left in (1). In the last section we will also use systems of *right language equations* of the form

$$\begin{aligned} X_i &= \bigcup_{j=1}^{m} F_{i,j}(X) \cdot a_j \cup \delta_i, \quad i = 1, \ldots, n, \\ X_0 &= G(X). \end{aligned} \tag{2}$$

In order to show that (1) has a unique solution we will first construct a larger system of equations, the *left quotient equations* defined by (1). We require the following result (see also [7]).

Proposition 1. *Let $X, Y \subseteq A^*$ be expressed in terms of their left quotients:*

$$X = \bigcup_{a \in A} a \cdot X_a \cup \delta_X, \qquad Y = \bigcup_{a \in A} a \cdot Y_a \cup \delta_Y.$$

Then

$$X \cup Y = \bigcup_{a \in A} a \cdot (X_a \cup Y_a) \cup (\delta_X \cup \delta_Y), \tag{3}$$

$$X \cap Y = \bigcup_{a \in A} a \cdot (X_a \cap Y_a) \cup (\delta_X \cap \delta_Y), \tag{4}$$

$$\bar{X} = \bigcup_{a \in A} a \cdot (\overline{X_a}) \cup (\lambda - \delta_X). \tag{5}$$

Proof. This is easily verified.

Given a set of left language equations

$$X_i = \bigcup_{j=1}^{m} a_j \cdot F_{i,j}(X) \cup \delta_i \tag{6}$$

as in (1), we construct the set

$$F_k = \bigcup_{j=1}^{m} a_j \cdot \tilde{F}_{k,j}(X) \cup \tilde{\delta}_k \tag{7}$$

where F_k ranges over the 2^{2^n} language functions, i.e. $k = 1, \ldots, 2^{2^n}$. This is done as follows: For each language function F_k in L_n write an expression involving $X_1, \ldots, X_n$, $\cup$, $\cap$, $^-$, $\emptyset$ and I. Compute the functions $\tilde{F}_{k,j}$ for F_k, $j = 1, \ldots, m$, by using (3), (4) and (5). We call (7) the *left quotient equations* generated by (6).

In our example we have

$$X_1 = a \cdot \bar{X}_2 \cup b \cdot I \quad \text{and} \quad X_2 = a \cdot X_2 \cup b \cdot (X_1 \cap \bar{X}_2) \cup \lambda.$$

We first note that

$$\emptyset = a \cdot \emptyset \cup b \cdot \emptyset \quad \text{and} \quad I = a \cdot I \cup b \cdot I \cup \lambda.$$

Next we can find the functions corresponding to $\bar{X}_i$

$$\bar{X}_1 = a \cdot X_2 \cup b \cdot \emptyset \cup \lambda, \qquad \bar{X}_2 = a \cdot \bar{X}_2 \cup b \cdot (\bar{X}_1 \cup X_2).$$

Similarly

$$X_1 \cap X_2 = a \cdot \emptyset \cup b \cdot (X_1 \cap \bar{X}_2), \qquad X_1 \cap \bar{X}_2 = a \cdot \bar{X}_2 \cup b \cdot (\bar{X}_1 \cup X_2),$$

etc. In this way we can construct a set of $2^{2^2} = 16$ equations of the form (7).

Theorem 1. *Any system of left language equations of the form* (6) *has a unique solution for each X_i, $i = 1, \ldots, n$. Furthermore each X_i is regular.*

Proof. The system (7) of equations generated by (6) as described above has the form of left quotient equations. Hence this system of equations can be solved using the fact that the equation

$$X = BX \cup C, \quad \lambda \notin B,$$

has the unique solution $X = B^*C$ which is regular if B and C are regular. This last result is due to Arden [1] and Bodnarchuk [2]; see also [7]. Thus we can find F_k for $k = 1, \ldots, 2^{2^n}$. Note however that each X_i represents one function in L_n, i.e. the left language equations (6) are contained in the list of left quotient equations (7). Hence we have a unique solution for $X_1, \ldots, X_n$ which satisfies (6) since it is part of the solution of (7).

Returning to our example note that we are interested in finding $X_0 = X_1 \cap X_2$. In general, it is not necessary to find all 16 left quotient equations; we need only those functions that are 'reachable' from X_0, i.e. only the quotients of X_0. Note also that the 16 distinct boolean functions of two variables in (7) do not necessarily define 16 distinct languages. In our example, the solution for X_1 and X_2 satisfies $X_1 \cap \bar{X}_2 = \bar{X}_2$, though $X_1 \cap \bar{X}_2$ and $\bar{X}_2$ denote different functions in L_n. However, the following system of equations generates 16 different languages:

$$X_1 = a \cdot (\bar{X}_1 \cup \bar{X}_2) \cup b \cdot X_2,$$
$$X_2 = a \cdot \bar{X}_1 \cup b \cdot ((\bar{X}_1 \cap \bar{X}_2) \cup (X_1 \cap X_2)).$$

The verification of this claim is straightforward and is left to the reader.

Clearly, Theorem 1 also holds for right language equations; for the proof we just have to replace left by right, and we have to interchange letters and variables on the right-hand side of the equations. We summarize:

Corollary 1. *Any system of right language equations of the form* (2) *has a unique solution for each X_i, $i = 1, \ldots, n$. Furthermore each X_i is regular.*

4. Boolean Automata

For the systems of equations (1) we can define a type of finite automaton where the 'next state' of a given state is not a set of states but a boolean function of the set of states. It was recently pointed out to us that the concepts in this section have been introduced by Kozen [6] in a different setting.

Formally a *boolean automaton* is a quintuple

$$\mathcal{B} = (A, Q, \tau, f^0, F),$$

where A is the *input alphabet*, $Q = \{q_1, \ldots, q_n\}$ is the finite, nonempty *set of states*, $\tau : Q \times A \to B_Q$ is the *transition function* which gives for each state and each letter a boolean function in B_Q, f^0 is the *initial function* in B_Q, and $F \subseteq Q$ is the set of final states. For example let $\mathcal{B} = (\{a, b\}, \{q_1, q_2\}, \tau, q_1 \wedge q_2, \{q_2\})$, where τ is given in Fig. 3.

	a	b
q_1	q_2'	1
q_2	q_2	$q_1 \wedge q_2'$

Fig. 3. A boolean automaton.

We extend the transition function τ to $B_Q \times A^*$ as follows. For $w \in A^*$, $a_j \in A$, $i = 1, \ldots, n$

$$\tau(q_i, \lambda) = q_i, \tag{8a}$$

$$\tau(q_i, a_j w) = f_{i,j}(\tau(q_1, w), \ldots, \tau(q_n, w)) \tag{8b}$$

where $f_{i,j} = \tau(q_i, a_j)$.

Now for any $f \in B_Q$ define

$$\tau(f, w) = f(\tau(q_1, w), \ldots, \tau(q_n, w)). \tag{9}$$

We remark, that we could replace (8b) by any of the following two definitions:

$$\tau(q_i, a_j w) = \tau(\tau(q_i, a_j), w) \quad \text{or} \quad \tau(q_i, w a_j) = \tau(\tau(q_i, w), a_j).$$

One can verify that all three ways of defining τ on $B_Q \times A^*$ yield the same function.

We now define acceptance of a word $w \in A^*$ by a boolean automaton $\mathcal{B}$. Let $h = \tau(f^0, w)$. Then

$$w \in L(\mathcal{B}) \quad \text{iff} \quad h(c_1, \ldots, c_n) = 1,$$

where $c_1 = 1$ if $q_i \in F$ and $c_i = 0$ otherwise.

To illustrate these concepts consider Fig. 3. We find

$$\begin{aligned}\tau(q_1, ab) &= q_2'(\tau(q_1, b), \tau(q_2, b)) = q_2'(1, q_1 \wedge q_2') = (q_1 \wedge q_2')',\\ \tau(q_2, ab) &= q_2(\tau(q_1, b), \tau(q_2, b)) = (q_1 \wedge q_2'),\\ \tau(f^0, ab) &= f^0(\tau(q_1, ab), \tau(q_2, ab))\\ &= f^0((q_1 \wedge q_2')', (q_1 \wedge q_2')) = (q_1 \wedge q_2')' \wedge (q_1 \wedge q_2') = 0.\end{aligned}$$

To determine whether $ab \in L(\mathscr{B})$ we now evalute $\tau(f^0, ab)$ at $(0, 1)$. Clearly we obtain 0. Hence $ab \notin L(\mathscr{B})$.

The following theorem summarizes the main property of boolean automata.

Theorem 2. *For every boolean automaton $\mathscr{B}$ with n states there exists an equivalent deterministic automaton $\mathscr{A}_{\mathscr{B}}$ with at most 2^{2^n} states, such that $L(\mathscr{A}_{\mathscr{B}}) = L(\mathscr{B})$.*

Proof. Construct the derived deterministic automaton

$$\mathscr{A}_{\mathscr{B}} = (A, P, \mu, f^0, G)$$

as follows:

$$P = \{\tau(f^0, w) \mid w \in A^*\},$$

$$G = \{f \in P \mid f(c_1, \ldots, c_n) = 1 \text{ for } c_i = 1 \text{ if } q_i \in F,\ c_i = 0 \text{ otherwise}\}$$

and

$$\mu(\tau(f^0, w), a) = \tau(f^0, wa) \quad \text{for all } w \in A^* \text{ and } a \in A.$$

Note that $P \subseteq B_Q$; hence $\mathscr{A}_{\mathscr{B}}$ is a deterministic finite automaton. Furthermore one verifies that $L(\mathscr{A}_{\mathscr{B}}) = L(\mathscr{B})$.

In summary, boolean automata or the corresponding language equations provide a potentially concise way of representing regular languages. Suppose the reduced deterministic automaton of a regular language L has n states. Then the nondeterministic automaton representation uses at least $\lceil \log_2 n \rceil$ states, whereas the boolean automaton representation uses at least $\lceil \log_2 \log_2 n \rceil$, where $\lceil a \rceil$ denotes the smallest integer $\geqslant a$.

Finally we remark that the language equations corresponding to boolean automata are 'as general as possible', if one wants their solutions to be regular, in the following sense. If one permits concatenation in the expressions for the $F_{i,j}$, the result need not be regular. For example, one verifies that

$$X = a(X \cdot Y) \cup \lambda, \qquad Y = b \cdot Z, \qquad Z = \lambda$$

has the unique solution $X = \{a^n b^n \mid n \geqslant 0\}$ which is not regular.

5. Sequential networks

In this section we will show how to use the methods developed in the previous sections to describe the language defined by a sequential network in a more direct way than was previously known. First we briefly describe our network and the language it defines. (For more technical details on the realization of sequential networks see, for example, [5].) Then we give a detailed example, contrasting our method with the classical approach, before we formally prove the relation between networks and equations.

We will consider sequential networks with 'decoded inputs' as shown in Fig. 4. The rectangles labelled Δ represent unit delays; the circles labelled $\wedge$, $\vee$, and $\sim$ represent AND gates, OR gates, and inverters, respectively. The inputs $x_{a_1}, \ldots, x_{a_m}$ are binary decoded inputs, $y_1, \ldots, y_n$ are the state variables, and z is the output. This type of network is a commonly used idealized model of sequential network operating synchronously. By 'decoded inputs' we mean the following: suppose we have an abstract alphabet A of m elements. These m elements are represented by $k = \lceil \log_2 m \rceil$ binary inputs $u_1, \ldots, u_k$. Each input x_{a_i} of $\mathcal{N}$ is obtained by a decoder from the u_l. More precisely $x_{a_i} = v_1 \wedge \cdots \wedge v_k$, where $v_j = u_j$ if the jth digit of the binary representation of $i-1$ is 1 and $v_j = u_j'$ otherwise. For example, let $k = 2$. Then we can have four decoded inputs:

$$x_0 = u_1' \wedge u_2', \qquad x_1 = u_1' \wedge u_2,$$

$$x_2 = u_1 \wedge u_2', \qquad x_3 = u_1 \wedge u_2.$$

Only one of the x_i will be 1 at any given time, and not all of them need be used.

In Fig. 4, $\hat{f}_{i,j} = f_{i,j} \wedge x_{a_j}$, where $f_{i,j}$ is a boolean function of $y_1, \ldots, y_n$ only, for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. The single binary output z of $\mathcal{N}$ is determined by g which is also a boolean function of $y = (y_1, \ldots, y_n)$. Clearly for any finite automaton $\mathcal{A}$ we can always find a network $\mathcal{N}$ realizing $\mathcal{A}$ and design it in the form shown in Fig. 4. The network $\mathcal{N}$ is now completely defined by its initial state and the next-state and output equations:

initial state: $\quad y^0 = (y_1^0, \ldots, y_n^0)$

next-state equations:

$$\begin{aligned} Y_1 &= (f_{1,1}(y) \wedge x_{a_1}) \vee \cdots \vee (f_{1,m}(y) \wedge x_{a_m}) \\ &\vdots \\ Y_n &= (f_{n,1}(y) \wedge x_{a_1}) \vee \cdots \vee (f_{n,m}(y) \wedge x_{a_m}) \end{aligned} \tag{10}$$

output equation: $\quad z = g(y).$

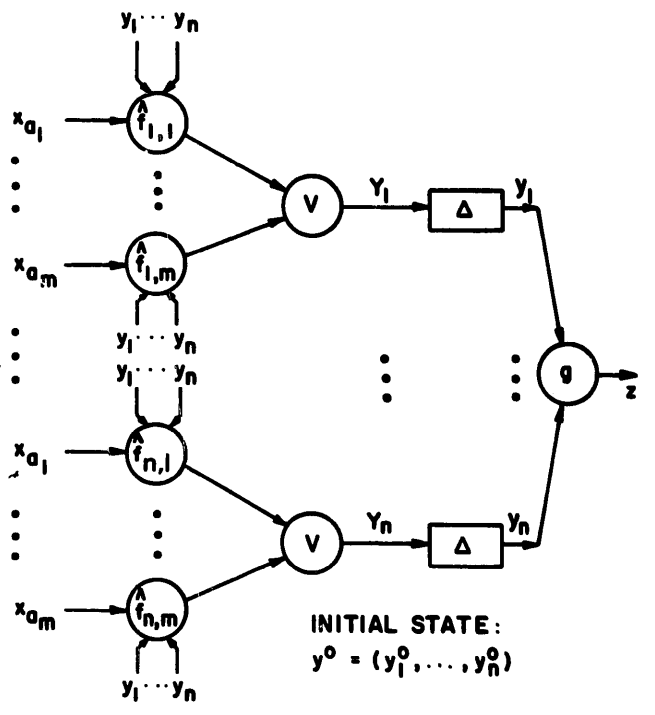

Fig. 4. General sequential network $\mathcal{N}$.

We now define the state y^w of $\mathcal{N}$ reached by $\mathcal{N}$ when we apply w to $\mathcal{N}$ started in y^0. This is done by induction on $|w|$,

$$\begin{aligned} y^{\lambda} &= y^0, \quad \text{the initial state,} \\ y^{wa_j} &= (f_{1,j}(y^w), \ldots, f_{n,j}(y^w)). \end{aligned} \tag{11}$$

Clearly this corresponds to the usual computation of the next state of the network using the next-state equations.

Next we define acceptance of a word $w \in A^*$ by $\mathcal{N}$ as follows:

$$w \in L(\mathcal{N}) \quad \text{iff} \quad g(y^w) = 1. \tag{12}$$

To illustrate these concepts consider the sequential network $\mathcal{N}_1$ given by Fig. 5. It is easily verified that $\mathcal{N}_1$ is described as follows:

$$\begin{array}{ll} \text{next-state equations:} & Y_1 = (y_2') \wedge x_a \vee (1) \wedge x_b, \\ & Y_2 = (y_2) \wedge x_a \vee (y_1 \wedge y_2') \wedge x_b, \\ \text{output equation:} & z = y_1 \wedge y_2, \\ \text{initial state:} & (y_1^0, y_2^0) = (0, 1). \end{array} \tag{13}$$

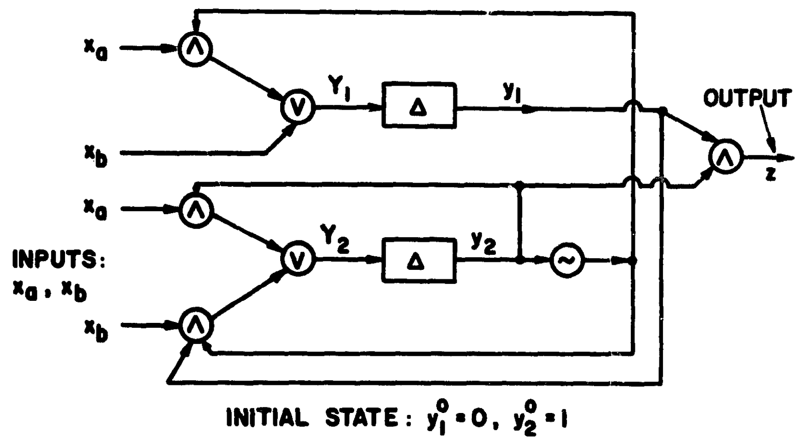

Fig. 5. The network $\mathcal{N}_1$.

Now, let $w = abb$. We have

$$\begin{aligned}
y^\lambda &= y^0 = (0, 1),\\
y^a &= ((y_2^\lambda)', y_2^\lambda) = (0, 1),\\
y^{ab} &= (1, y_1^a \wedge (y_2^a)') = (1, 0 \wedge 1') = (1, 0),\\
y^{abb} &= (1, y_1^{ab} \wedge (y_2^{ab})') = (1, 1 \wedge 1) = (1, 1).
\end{aligned}$$

Since $g(y^{abb}) = 1$, we conclude $abb \in L(\mathcal{N}_1)$.

Let us illustrate the classical method of describing the language accepted by a network with the aid of our example. We obtain the transition table as shown in Fig. 6(a), where the arrow points to the initial state. Note that the state 00 is not reachable from the initial state; hence a more appropriate table is that of Fig. 6(b) which shows only the reachable states. Note that the output z depends only on the state; its value is shown in the rightmost column of the table.

y_1	y_2	x_a		x_b		z
0	0	1	0	1	0	0
→0	1	0	1	1	0	0
1	0	1	0	1	1	0
1	1	0	1	1	0	1
		Y_1		Y_2		

(a)

y_1	y_2	x_a		x_b		z
→0	1	0	1	1	0	0
1	0	1	0	1	1	0
1	1	0	1	1	0	1
		Y_1		Y_2		

(b)

Fig. 6. Transition table of $\mathcal{N}_1$.

We have now in Fig. 6(b) the finite automaton defined by the network $\mathcal{N}_1$, if we interpret $z = 1$ as denoting an accepting state. The state graph of this automaton is shown in Fig. 7, where we have 1, 2 and 3 instead of 01, 10 and 11 as state symbols. One can verify that $\mathcal{A}_1$ is reduced.

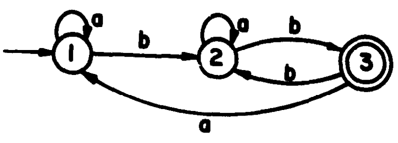

Figure 7. Finite automaton $\mathscr{A}_1$ defined by $\mathscr{N}_1$.

In this rather indirect way we have reached a point where we have defined the language $L(\mathscr{N}_1)$. This language can be specified in many ways. For instance, we can interpret the automaton $\mathscr{A}_1$ as a satisfactory description of $L = L(\mathscr{N}_1) = L(\mathscr{A}_1)$, or we can write a regular expression for L, e.g.

$$L = a^*b(a \cup ba^*b)^*b.$$

We now proceed to show that the language $L = L(\mathscr{N})$ can be related much more closely to $\mathscr{N}$ than in the classical approach described above. Suppose we translate the set (13) of equations to the following system of right language equations:

$$\begin{aligned} X_1 &= \bar{X}_2 \cdot a \cup I \cdot b, \\ X_2 &= X_2 \cdot a \cup (X_1 \cap \bar{X}_2) \cdot b \cup \lambda, \qquad (14) \\ X_0 &= X_1 \cap X_2. \end{aligned}$$

The motivation for this is as follows. Let X_i be the set of all words in A^* that lead $\mathscr{N}_1$ to a state with $y_i = 1$, when started in y_1^0, y_2^0, for $i = 1, 2$. Suppose now that $w \in A^*$ and $w \notin X_2$, i.e. $y_2 = 0$ after w is applied. From the network $\mathscr{N}_1$ it is clear that $y_2' = 1$ and, if $x_a = 1$, $Y_1 = 1$. Thus the word wa will result in a state where $y_1 = 1$. We conclude that $X_1 \supseteq \overline{X_2}a$. The remaining pieces of (14) are similarly obtained. Note that we have used all the information about $\mathscr{N}_1$ to write (14). In fact the next-state equations define the non-empty words of X_1 and X_2; the initial state determines whether or not $\lambda \in X_i$, $i = 1, 2$; and the output equation leads to the 'output language' equation for X_0.

According to Corollary 1, (14) has a unique solution for X_0, and, in fact, one can verify that $X_0 = L(\mathscr{N}_1) = L(\mathscr{A}_1)$. In general, we will give a direct translation of next-state equations, output equation and initial state of a sequential network $\mathscr{N}$ which leads to the language $L(\mathscr{N})$. We first define this transformation.

Given the description (10) of a sequential network with decoded inputs we derive a system of right language equations defined by

$$\begin{aligned} X_1 &= F_{1,1}(X) \cdot a_1 \cup \cdots \cup F_{1,m}(X) \cdot a_m \cup \delta_1 \\ &\vdots \\ X_n &= F_{n,1}(X) \cdot a_1 \cup \cdots \cup F_{n,m}(X) \cdot a_m \cup \delta_n \qquad (15) \\ X_0 &= G(X) \end{aligned}$$

where $X = X_1, \ldots, X_n$; $\delta_i = \lambda$ if $y_i^0 = 1$ and $\delta_i = \emptyset$ otherwise; $F_{i,j}$ is the language function in L_n which corresponds to $f_{i,j}$ for $i = 1, \ldots, n, j = 1, \ldots, m$; and G is the language function in L_n corresponding to g.

Note that the $\wedge$ function of (10) in the expression $f_{i,j}(y) \wedge x_{a_j}$ is replaced by concatenation in (15).

We now have to show that the unique, regular solution of (15) is the language accepted by the network given by (10), i.e. that

$$X_0 = L(\mathcal{N}).$$

Lemma 1. *Let $f \in B_n$ be a boolean function and let F be the corresponding function in L_n. For $i = 1, \ldots, n$ let $y_i \in \{0, 1\}$ and $X_i \subseteq A^*$ be such that $y_i = 1$ iff $\lambda \in X_i$. Then*

$$f(y) = 1 \quad \textit{iff} \quad \lambda \in F(X). \tag{16}$$

Proof. Assume that the function f is represented by some standard expression in the symbols 0, 1, $y_1, \ldots, y_n$ and operators $\vee$, $\wedge$ and $'$. We proceed by structural induction on the number r of operators in that expression.

Basis, $r = 0$. (a) If $f = 0$, then $F = \emptyset$ and (16) holds.

(b) If $f = 1$, then $F = I$ and (16) holds.

(c) If $f = y_i$ for some $i \in \{1, \ldots, n\}$, then $F = X_i$. By assumption $y_i = 1$ iff $\lambda \in X_i$. Hence (16) holds.

Induction step, $r > 0$. Assume now that (16) holds for g and G, as well as for h and H:

(a) $f = g \vee h$, $F = G \cup H$,

$$\begin{aligned} f(y) = 1 \quad &\text{iff} \quad g(y) \vee h(y) = 1 \\ &\text{iff} \quad g(y) = 1 \text{ or } h(y) = 1 \\ &\text{iff} \quad \lambda \in G(X) \quad \text{or} \quad \lambda \in H(X) \\ &\text{iff} \quad \lambda \in G(X) \cup H(X) = F(X). \end{aligned}$$

(b) $f = g \wedge h$, $F = G \cap H$. The proof is similar to (a).

(c) $f = g'$, $F = \bar{G}$. This case follows easily. Thus the induction step holds.

This result shows that $L(\mathcal{N})$ and X_0 agree as far as the empty word is concerned, for

$$\lambda \in L(\mathcal{N}) \quad \text{iff} \quad g(y^0) = 1.$$

By construction of (15), $y_i^0 = 1$ iff $\lambda \in X_i$ for $i = 1, \ldots, n$. Thus Lemma 1 applies and $g(y^0) = 1$ iff $\lambda \in G(X)$. Next we would like to show that $w \in L(\mathcal{N})$ iff $w \in X_0$ by induction on the length of w. We encounter the following problem. Let $w \in A^*$. For the network we must compute y^w inductively as in (11), i.e.

$$y_i^\lambda = y_i^0$$

and

$$y_i^{wa_j} = f_{i,j}(y^w).$$

Note that in this computation the letters of w are used from *left to right*. On the other hand the computation of right quotients of the X_i involves the use of the letters of w from *right to left*, for we have

$$X/\lambda = X$$

and

$$X/a_j w = (X/w)/a_j.$$

In view of these difficulties it is convenient to reverse the system (15) of equations. To simplify the notation we will let $V_{i,j} = X_{i,j}^\rho$, $V_i = X_i^\rho$, and $V = V_1, \ldots, V_n$. This formal reversal yields the system of left language equations:

$$\begin{aligned} V_i &= a_1 \cdot F_{i,1}(V) \cup \cdots \cup a_m \cdot F_{i,m}(V) \cup \delta_i, \quad i = 1, \ldots, n, \\ V_0 &= G(V). \end{aligned} \tag{17}$$

Now we can deal with the left quotients of V.

Proposition 2. *Let $X = X_1, \ldots, X_n$ be languages and F a language function in L_n. Then for all $w \in A^*$*

$$w \backslash F(X) = F(w \backslash X).$$

Proof. The proof follows easily by structural induction on F. It is sufficient to verify that

$$w \backslash (G(X) \cup H(X)) = w \backslash G(X) \cup w \backslash H(X),$$

and

$$w \backslash \bar{G}(X) = \overline{w \backslash G(X)}.$$

This follows from the definition of left quotients.

Lemma 2. *In the system* (17) *of left language equations, for all $i = 1, \ldots, n$, $a_j \in A$, $w \in A^*$*

$$a_j w \backslash V_i = F_{i,j}(w \backslash V). \tag{18}$$

Proof. We proceed by induction on $r = |w|$.
Basis, $r = 0$. Here $w = \lambda$, and (18) reduces to:

$$a_j \backslash V_i = F_{i,j}(V).$$

This follows immediately from (17).

Induction step, $r > 0$. Assume (18) holds for wa_k, where $a_k \in A$. Then

$$a_j w a_k \backslash V_i = a_k \backslash (a_j w \backslash V_i) = a_k \backslash F_{i,j}(w \backslash V) = F_{i,j}(w a_k \backslash V),$$

by Proposition 2.

Lemma 3. *Let $w \in A^*$, let y^w be defined by* (11) *and let V be as in* (17). *Then for all $i = 1, \ldots, n$*

$$y_i^w = 1 \quad \textit{iff} \quad \lambda \in w^\rho \backslash V_i.$$

Proof. We proceed by induction on $r = |w|$.
Basis, $r = 0$. Here $w = \lambda$, $y_i^\lambda = y_i^0$ and $\lambda^\rho \backslash V_i = V_i$. By construction of (17), $y_i^0 = 1$ iff $\lambda \in V_i$.
Induction step, $r > 0$. Assume the result holds for w and let $a_j \in A$. Then by (11)

$$y_i^{wa_j} = f_{i,j}(y^w).$$

By Lemma 2

$$a_j w^\rho \backslash V_i = F_{i,j}(w^\rho \backslash V).$$

By the inductive assumption $y_k^w = 1$ iff $\lambda \in w^\rho \backslash V_k$ for all $k = 1, \ldots, n$. By Lemma 1, $f_{i,j}(y^w) = 1$ iff $\lambda \in F_{i,j}(w^\rho \backslash V)$. Note that $a_j w^\rho = (wa_j)^\rho$. Thus $\lambda \in (wa_j)^\rho \backslash V_i$ iff $y^{wa_j} = 1$ as required.

We now prove our main result about networks:

Theorem 3. *Let $\mathcal{N}$ be a network defined by* (10) *and let X_0 be the solution of the corresponding system of right language equations* (15). *Then $L(\mathcal{N}) = X_0$.*

Proof. By the definition of acceptance (12) we have

$$w \in L(\mathcal{N}) \quad \text{iff} \quad g(y^w) = 1.$$

Because of Lemma 3, we can apply Lemma 1 to y^w and

$$\begin{aligned} g(y^w) = 1 \quad &\text{iff} \quad \lambda \in G(w^\rho \backslash V) \\ &\text{iff} \quad \lambda \in G(w^\rho \backslash X^\rho) \\ &\text{iff} \quad \lambda \in G((w^\rho \backslash X^\rho)^\rho) \\ &\text{iff} \quad \lambda \in G(X/w) \\ &\text{iff} \quad \lambda \in G(X)/w \\ &\text{iff} \quad w \in G(X) = X_0 \end{aligned}$$

where we have used the (easily verified) property $(w^\rho \backslash X^\rho)^\rho = X/w$.

We now relate boolean automata to networks.

Theorem 4. *For every boolean automaton $\mathcal{B} = (A, Q, \tau, f^0, F)$ with n states there exists a sequential network $\mathcal{N}$ with n unit delays such that $L(\mathcal{N}) = [L(\mathcal{B})]^\rho$. Conversely,* *for every sequential network $\mathcal{N}$ with n unit delays there exists a boolean automaton $\mathcal{B}$ with n states such that $L(\mathcal{B}) = [L(\mathcal{N})]^\rho$.*

Proof. Let $\mathcal{B} = (A, Q, \tau, f^0, F)$ be a boolean automaton with $Q = \{q_1, \ldots, q_n\}$. Define a system $\mathcal{E}$ of left language equations derived from $\mathcal{B}$ as follows:

$$X_i = \bigcup_{a \in A} a \cdot F_{i,a} \cup \delta_i, \quad i = 1, \ldots, n,$$

$$X_0 = F^0$$

where $F_{i,a} \in L_n$ corresponds to $\tau(q_i, a)$ for $i = 1, \ldots, n$ and $a \in A$ via the isomorphism between B_Q and L_n, and F^0 corresponds to f^0 in a similar way. Also $\delta_i = \lambda$ if $q_i \in F$ and $\delta_i = \emptyset$ otherwise.

We will show that $L(\mathcal{B}) = L(\mathcal{E})$. First we note that

$$\begin{aligned} w \in L(\mathcal{E}) \quad &\text{iff} \quad w \in X_0 = F^0(X) \\ &\text{iff} \quad \lambda \in w \backslash F^0(X) = F^0(w \backslash X) = F^0(w \backslash X_1, \ldots, w \backslash X_n), \end{aligned}$$

or

$$w \in L(\mathcal{E}) \quad \text{iff} \quad \lambda \in F^0(w \backslash X_1, \ldots, w \backslash X_n). \tag{19}$$

Secondly, let $\tau(q_i, w) = q_i^w$ for $i = 1, \ldots, n$ and $w \in A^*$, and let $c = c_1, \ldots, c_n$ where

$$c_i = \begin{cases} 0, & \text{if } q_i \notin F, \\ 1, & \text{otherwise.} \end{cases}$$

We have

$$\begin{aligned} w \in L(\mathcal{B}) \quad &\text{iff} \quad [\tau(f^0, w)](c) = 1 \\ &\text{iff} \quad [f^0(\tau(q_1, w), \ldots, \tau(q_n, w))](c) = 1 \\ &\text{iff} \quad [f^0(q_1^w, \ldots, q_n^w)](c) = 1 \\ &\text{iff} \quad f^0(q_1^w(c), \ldots, q_n^w(c)) = 1, \end{aligned}$$

or

$$w \in L(\mathcal{B}) \quad \text{iff} \quad f^0(q_1^w(c), \ldots, q_n^w(c)) = 1. \tag{20}$$

Now we claim that for all $i = 1, \ldots, n$ and all $w \in A^*$,

$$\lambda \in w \backslash X_i \quad \text{iff} \quad q_i^w(c) = 1. \tag{21}$$

If we assume this claim, then Lemma 1 applies and

$$f^0(q_1^w(c), \ldots, q_n^w(c)) = 1 \quad \text{iff} \quad \lambda \in F^0(w \backslash X_1, \ldots, w \backslash X_n),$$

thus showing that $L(\mathcal{B}) = L(\mathcal{E})$ by (19) and (20).

The proof of (21) follows by induction on $|w|$. For $w = \lambda$ we have $\lambda \in \lambda \backslash X_i = X_i$ iff $q_i \in F$ iff $c_i = 1$, by construction. Now assume the claim holds for w and consider $a_j w$. By Lemma 2, the induction hypothesis, and Lemma 1

$$\begin{aligned} \lambda \in a_j w \backslash X_i \quad &\text{iff} \quad \lambda \in F_{i,j}(w\backslash X_1, \ldots, w\backslash X_n) \\ &\text{iff} \quad f_{i,j}(q_1^w(c), \ldots, q_n^w(c)) = 1. \end{aligned}$$

But

$$q_i^{a_j w} = \tau(q_i, a_j w) = f_{i,j}(q_1^w, \ldots, q_n^w).$$

Hence the induction step goes through, and (21) holds.

Now we construct a network $\mathcal{N}$ so that its language will satisfy the reverse of $\mathscr{E}$. Namely, the network equations are of the form:

$$\begin{aligned} &Y_i = (f_{i,1}(y) \wedge x_{a_1}) \vee \cdots \vee (f_{i,m}(y) \wedge x_{a_m}), \quad i = 1, \ldots, n. \\ &y^0 = c, \\ &z = f^0(y). \end{aligned}$$

It is then clear that the equations for $L(\mathcal{N})$ are obtained by reversing $\mathscr{E}$. Hence $[L(\mathcal{N})]^\rho = L(\mathscr{E}) = L(\mathscr{B})$. Thus we have constructed $\mathcal{N}$ from $\mathscr{B}$ in such a way that $L(\mathcal{N}) = [L(\mathscr{B})]^\rho$.

Reversing the argument proves the converse claim.

References

[1] D.N. Arden, Delayed logic and finite state machines, *Proc. 2nd Ann. Symp. on Switching Circuit Theory and Logical Design,* (Detroit, 1961) 133–151.

[2] K. Bodnarchuk, Sistemy uravnenij v algebre sobytij, *Z. Vyčisl. Mat. i Fiz.* **3** (1963) 1077–1088.

[3] J.A. Brzozowski, Canonical regular expressions and minimal state graphs for definite events, in: *Mathematical Theory of Automata,* (Polytechnic Institute of Brooklyn, New York, 1963) 529–561.

[4] J.A. Brzozowski, Regular expressions from sequential circuits, *IEEE Trans.* **EC-13** (6) (1964) 741–744.

[5] J.A. Brzozowski and M. Yoeli, *Digital Networks* (Prentice Hall, Englewood Cliffs, NJ, 1976).

[6] D. Kozen, On Parallelism in Turing machines, *Proc. 17th Ann. Symp. on Foundations of Computer Science* (Polytechnic Institute of Brooklyn, 1976) 89–97.

[7] A. Salomaa, *Theory of Automata* (Pergamon Press, Oxford, 1969).