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

by

J.A. Brzozowski and E. Leiss
Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

Research Report CS-77-31

October 1977

This research was supported by the National Research Council of Canada under grant No. A-1617.

On Equations for Sequential Networks, Finite Automata, and Regular Languages

by

J.A. Brzozowski and E. Leiss

Abstract

We present a new method of determining the language $L(N)$ accepted by a sequential network N. First we derive a system E of equations of the form

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

where A is the alphabet of input symbols, the X_i are variables, n is the number of unit delays in N, the $F_{i,a}$ are boolean functions in the variables X_i, and δ_i is either the empty set or the empty word. The system E is in one-to-one correspondence with the next-state equations of N. We prove that E has a unique solution $L(E)$, show how to determine $L(E)$, and verify that $L(E) = L(N)$. The method of determining the solution $L(E)$ gives rise to a generalization of finite automata, called boolean automata. These automata provide a very concise representation of regular languages.

0. 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 ϕ, and I is a shorthand for the language A^*. The concatenation of two languages L_1 and L_2 is $L_1 \cdot L_2 = \{u \cdot v \mid u \in L_1, v \in L_2\}$. 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 $\{w^\rho \mid w \in L\}$.

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

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

Another finite boolean algebra which will be used is the set B_X of boolean functions $f : \overset{n}{\underset{i=1}{\times}} \{0,1\} \to \{0,1\}$ 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.

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

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

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, $\mathcal{A}$ is called deterministic. The function τ is extended to $P(Q) \times A^*$ in the usual way. We assume that $\mathcal{A}$ 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 $\mathcal{A}$ iff $\tau(Q_0,w) \cap F \neq \phi$. $L(\mathcal{A})$, the set of words accepted by $\mathcal{A}$, 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 $\mathcal{A} = (A, Q, M, q_0, F)$ is defined as follows: Let $Q_w = \{q \in Q \mid \tau(q, w) \in F\}$. Then $\mathcal{A}^\rho = (A, P, N, p_0, G)$, where

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

$$p_0 = F ,$$

$$G = \{p \in P \mid q_0 \in p\} , \text{ and}$$

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

$\mathcal{A}^\rho$ 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 [1]).

1. Introduction

It is well known that every sequential network accepts a regular language. A frequently used method of finding a deterministic automaton or a regular expression for this language is illustrated in Section 2. This method is rather indirect.

In this paper we present a direct method of relating networks to languages. From the next-state equations, the initial state, and the output logic of the network one derives a system of equations of the form

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

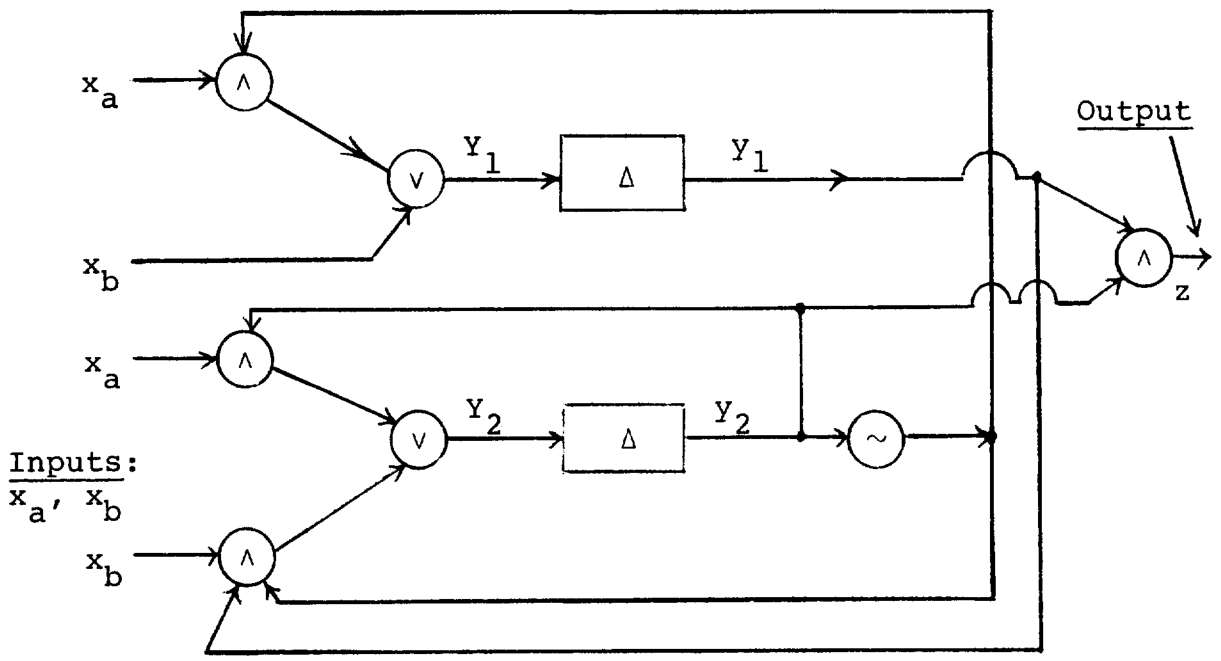
where the $X_1,\ldots,X_n$ are variables, the $F_{i,a}$ are language functions in the variables $X_1,\ldots,X_n$ (elements of L_n) , and $\delta_i \in \{\phi, \{\lambda\}\}$ for $i = 1,\ldots,n$. Conversely, any such system of equations uniquely determines a sequential network. We then show that such a system always has a unique solution, and give a method of determining it. We also prove that this solution is precisely the language accepted by the network. The approach gives rise to a generalization of the concept of finite automaton; namely, we introduce boolean automata, which provide a very concise representation of regular languages.

A method of obtaining a regular language from a given sequential network has been given in [2]; however, the basic approach used there was different.

2. An Example

We begin with a detailed example, contrasting our method with the classical approach.

Consider the sequential network of Figure 1. The rectangles labelled Δ represent unit delays. The circles



initial state: $y_1^0 = 0$, $y_2^0 = 1$

Figure 1 Network N

labelled $\wedge$, $\vee$, and $\sim$ represent AND gates, OR gates and inverters, respectively. The inputs x_a and x_b are binary inputs, y_1 and y_2 are the state variables, and z is the output. This network is an example of a commonly used idealized model of sequential network operating synchronously.

For more technical details on the realization of sequential networks see, for example, [3].

We will introduce a special assumption about the inputs x_a and x_b ; namely, we assume that the abstract input alphabet is $A = \{a, b\}$ and the letters are represented as follows. If $x_a = 1$ and $x_b = 0$ then the input applied to the network is a ; if $x_a = 0$ and $x_b = 1$ - the input is b . We assume that the remaining two combinations in which $x_a = x_b$ are not allowed. We will call such inputs x_a and x_b <u>decoded inputs</u>. In our example, one binary input u can be used (with $u = x_a$ and $u' = x_b$) to represent the alphabet $A = \{a, b\}$. The reasons for using decoded inputs will become clearer later. We also assume that the initial state is part of the network description.

Sequential networks of the type shown in Figure 1 can be analyzed as follows. We can first compute the next-state variables Y_1 and Y_2 and the output with the aid of the next-state and output equations:

$$\left.\begin{aligned} Y_1 &= (y_2') \wedge x_a \vee (1) \wedge x_b \\ Y_2 &= (y_2) \wedge x_a \vee (y_1 \wedge y_2') \wedge x_b \\ z &= y_1 \wedge y_2 \end{aligned}\right\} \qquad (1)$$

(The reasons for the parentheses will be explained a little later. We assume that $\wedge$ has precedence over $\vee$.) Next we can obtain the transition table as shown in Figure 2(a), where the arrow points to the initial state. Note that state

	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)

Figure 2 Transition table of N

00 is not reachable from the initial state; hence a more appropriate table is that of Figure 2(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.

We have now in Figure 2(b) the finite automaton defined by the network of Figure 1, if we interpret $z = 1$ as denoting an accepting state. The state graph of this automaton is shown in Figure 3, where the double circle denotes the accepting state, and where we have used 1, 2 and 3 instead of 01, 10 and 11 as state symbols. One can verify that A is reduced.

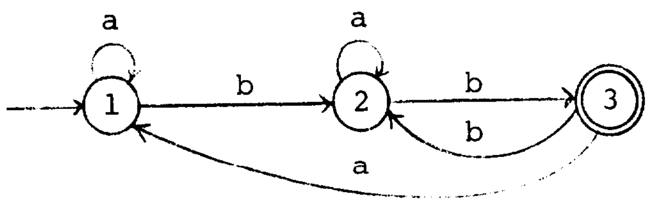


Figure 3 Finite automaton A defined by N

In this rather indirect way we have reached a point where we have defined the language $L(N)$ accepted by the network N of Figure 1. This language can be specified in many ways. For instance, we can interpret the automaton A as a satisfactory description of $L = L(N) = L(A)$, or we can write a regular expression for L , e.g.

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

We now proceed to show that the language $L = L(N)$ can be related much more closely to N than in the classical approach described above. Suppose we translate the set (1) of equations to the following system of right language equations:

$$\left.\begin{aligned} X_1 &= \bar{X}_2\cdot a \cup I\cdot b \\ X_2 &= X_2\cdot a \cup (X_1 \cap \bar{X}_2)\cdot b \cup \lambda \\ X_0 &= X_1 \cap X_2 \end{aligned}\right\} \qquad (2)$$

The motivation for this is as follows. Let X_i be the set of all words in A^* that lead N to a state with $y_i = 1$, when started in y_1^o, y_2^o , 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 of Figure 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 $X_1 \supseteq \bar{X}_2a$. The remaining pieces of (2) are similarly obtained. Note that we have used all the information about N to write (2). 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 .

One of the main results proved in this paper is that (2) has a unique solution for X_0 , and in fact $X_0 = L(N) = L(A)$, where N and A are defined in Figures 1 and 3, respectively. Thus a direct translation of the next-state and output equations of a sequential network N to a system of language equations leads to the language $L(N)$. We now verify our claim for the example.

From (2) we can obtain the right quotient equations for X_0 as shown below:

$$\left.\begin{aligned} X_0 = X_1 \cap X_2 &= \phi\cdot a \cup (X_1 \cap \bar{X}_2)\cdot b \\ \phi &= \phi\cdot a \cup \phi\cdot b \\ X_1 \cap \bar{X}_2 &= \bar{X}_2\cdot a \cup (\bar{X}_1 \cup X_2)\cdot b \\ \bar{X}_2 &= \bar{X}_2\cdot a \cup (\bar{X}_1 \cup X_2)\cdot b \\ \bar{X}_1 \cup X_2 &= X_2\cdot a \cup (X_1 \cap \bar{X}_2)\cdot b \cup \lambda \\ X_2 &= X_2\cdot a \cup (X_1 \cap \bar{X}_2)\cdot b \cup \lambda \end{aligned}\right\} \quad (3)$$

Noting that $X_1 \cap \bar{X}_2 = \bar{X}_2$ and $\bar{X}_1 \cup X_2 = X_2$ we have:

$$\left.\begin{aligned} X_0 = X_1 \cap X_2 &= \phi\cdot a \cup \bar{X}_2\cdot b \\ \phi &= \phi\cdot a \cup \phi\cdot b \\ \bar{X}_2 &= \bar{X}_2\cdot a \cup X_2\cdot b \\ X_2 &= X_2\cdot a \cup \bar{X}_2\cdot b \cup \lambda \end{aligned}\right\} \quad (4)$$

If we reverse (4) we have the derivative equations for X_0^ρ :

$$\left.\begin{array}{ll} 1 & X_0^\rho = a\cdot\phi \cup b\cdot\overline{X_2^\rho} \\ 2 & \phi = a\cdot\phi \cup b\cdot\phi \\ 3 & \overline{X_2^\rho} = a\cdot\overline{X_2^\rho} \cup b\cdot X_2^\rho \\ 4 & X_2^\rho = a\cdot X_2^\rho \cup b\cdot\overline{X_2^\rho} \cup \lambda \end{array}\right\} \qquad (5)$$

For convenience let us number the derivatives of X^ρ as shown in (5). Then (5) defines the state table of Figure 4.

	a	b	
→ 1	2	3	0
2	2	2	0
3	3	4	0
4	4	3	1

Figure 4 State table of finite automaton for X_0^ρ

To get the reduced finite automaton for X_0 we can now reverse the automaton of Figure 4 by the "subset construction" [1]. We obtain Figure 5. One verifies that this automaton is

	a	b	
→ {4}	{4}	{3}	0
{3}	{3}	{1,4}	0
{1,4}	{4}	{3}	1

Figure 5 Reduced automaton for X

isomorphic to the automaton of Figure 3 with the correspondence:

$$1 \leftrightarrow \{4\}$$
$$2 \leftrightarrow \{3\}$$
$$3 \leftrightarrow \{1,4\} \quad .$$

Thus we have demonstrated that $X_0 = L(N)$.

Alternatively we can proceed as follows. The system (4) can be solved for X_0 using the fact that

$$X = XU \cup V \ , \ \lambda \notin U \ \text{implies} \ X = VU^* \quad ,$$

for all languages U and V . This yields a regular expression for X_0 - e.g. $X_0 = (a \cup ba^*b)^*ba^*b$ - from which an automaton can be obtained.

In the sequel we prove that these ideas hold in general.

3. The Language Defined by a Network

The general form of a sequential network $\mathcal{N}$ with decoded inputs is shown in Figure 6. Suppose we have an abstract alphabet A of m elements. These m elements are represented by $k = \lceil \log_2 m \rceil$ binary inputs $u_1,\ldots,u_k$, where $\lceil x \rceil$ is the smallest integer $\geq x$. Each input x_{a_i} of $\mathcal{N}$ is obtained by a decoder from the u_ℓ. More precisely $x_{a_i} = v_1 \wedge \ldots \wedge v_k$, where $v_j = u_j$ if the j-th 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:

$$\begin{aligned} x_0 &= u_1' \wedge u_2' \\ x_1 &= u_1' \wedge u_2 \\ x_2 &= u_1 \wedge u_2' \\ x_3 &= u_1 \wedge u_2 \quad . \end{aligned}$$

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

In Figure 6, $\hat{f}_{i,j} = f_{i,j} \wedge x_{a_j}$, 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 Figure 6. The network $\mathcal{N}$ is now completely defined by its initial state and the next-state and output

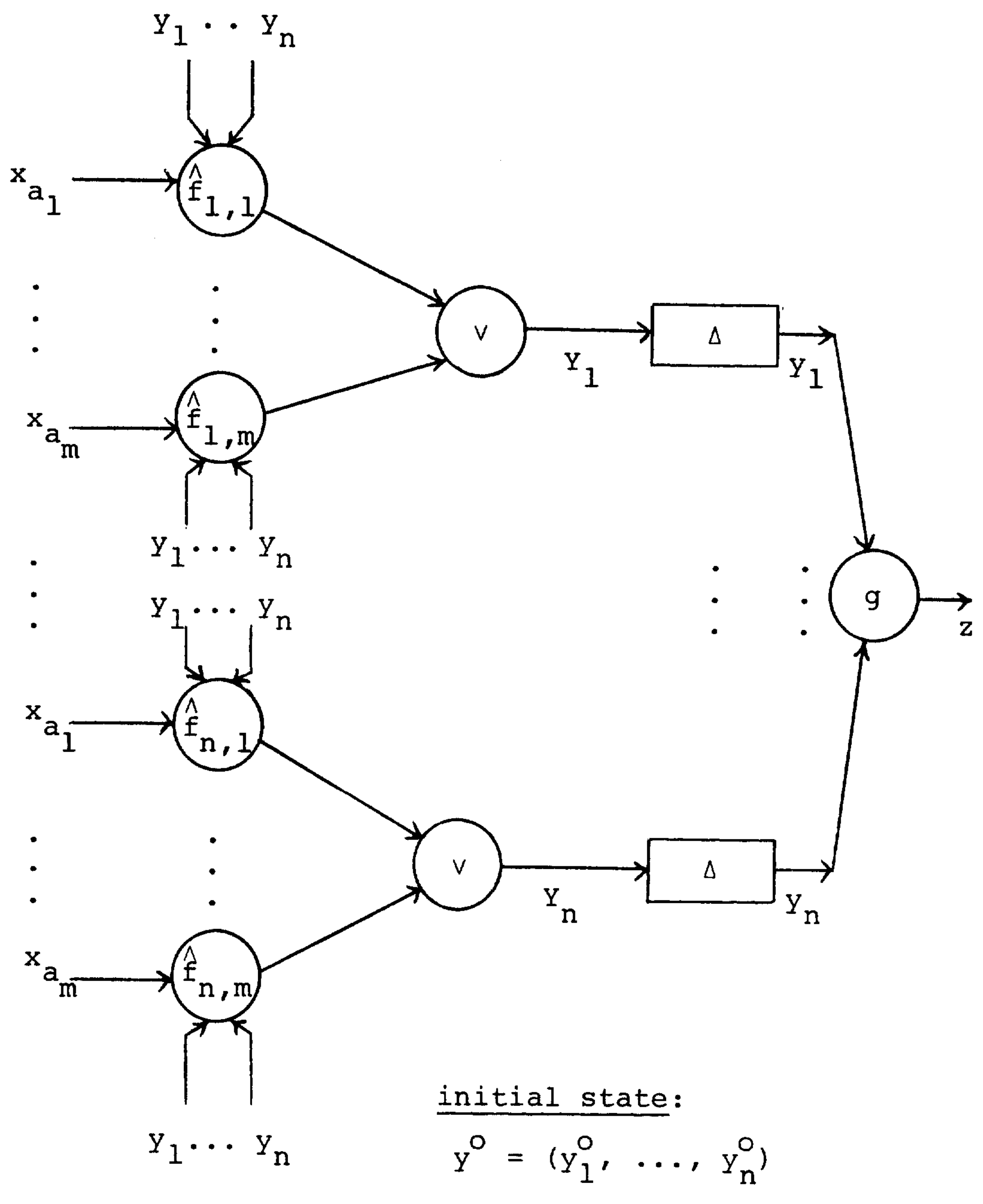


Figure 6 General sequential network

equations:

$$\left.\begin{array}{l}
\text{initial state:}\quad y^o = (y_1^o, \ldots, y_n^o) \\
\text{next-state equations:} \\
Y_1 = (f_{1,1}(y) \wedge x_{a_1}) \vee \ldots \vee (f_{1,m}(y) \wedge x_{a_m}) \\
\cdot \\
\cdot \\
\cdot \\
Y_n = (f_{n,1}(y) \wedge x_{a_1}) \vee \ldots \vee (f_{n,m}(y) \wedge x_{a_m}) \\
\text{output equation:} \\
z = g(y)
\end{array}\right\} \qquad (6)$$

We now define the state y^w of N reached by N when we apply w to N started in y^o. This is done by induction on $|w|$.

$$\left.\begin{array}{l}
y^\lambda = y^o \text{, the initial state} \\
y^{wa_j} = (f_{1,j}(y^w), \ldots, f_{n,j}(y^w))
\end{array}\right\} \qquad (7)$$

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 N as follows:

$$w \in L(N) \quad \text{iff} \quad g(y^w) = 1 . \qquad (8)$$

To illustrate this approach return to (1) and let $w = abb$. We have

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

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

4. Right Language Equations of a Network

Following the motivation of Section 2, we transform (6) into a system of equations, called right language equations, which has the form:

$$\left.\begin{array}{l} X_1 = F_{1,1}(X)\cdot a_1 \cup \ldots \cup F_{1,m}(X)\cdot a_m \cup \delta_1 \\ \cdot \\ \cdot \\ \cdot \\ X_n = F_{n,1}(X)\cdot a_1 \cup \ldots \cup F_{n,m}(X)\cdot a_m \cup \delta_n \\ X_0 = G(X) \end{array}\right\} \qquad (9)$$

where $X = X_1, \ldots, X_n$;

$\delta_i = \lambda$ if $y_i^o = 1$ and $\delta_i = \phi$ otherwise;

$F_{i,j}$ is a language function in L_n , derived from $f_{i,j}$ as shown below for $i = 1,\ldots,n$, and $j = 1,\ldots,m$;

G is a language function in L_n , derived from g as shown below.

The term "right language equations" reflects the fact that the letters a_i of the alphabet appear on the right in (9). Note that the AND function of (6) in the expression $f_{i,j}(y) \wedge x_{a_j}$ is replaced by concatenation in (9). We could also construct a system of left language equations of the form:

$$\left.\begin{array}{l} X_i = \bigcup_{j=1}^{m} a_j\cdot F_{i,j}(X) \cup \delta_i \quad i = 1,\ldots,n , \\ X_0 = G(X) \end{array}\right\} \qquad (10)$$

but this is not the concept that we need presently.

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

	B_n	L_n
constant functions:	0	ϕ
	1	$I=A^*$
variables:	$y = y_1,\ldots,y_n$	$X = X_1,\ldots,X_n$
operators:	$\wedge$	$\cap$
	$\vee$	$\cup$
	$'$	$\overline{}$
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, ..., y_n, $\wedge$, $\vee$ and $'$. In this expression replace y_i by X_i , $i = 1,\ldots,n$, 0 by ϕ , 1 by I , $\wedge$ by $\cap$, $\vee$ by $\cup$, and $'$ by $\overline{}$. We now have an expression in the variables X_1, ..., X_n . The language 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. To illustrate, we have $f_{2,b}$ given by the expression $y_1 \wedge y_2'$ in (1). Thus $F_{2,b} = X_1 \cap \overline{X_2}$. Compare (1) with (2).

In order to show that (9) has a unique solution we will first construct a larger system of equations, the <u>right quotient equations</u> defined by (9). We require the following result.

<u>Proposition 1</u> Let $X, Y \subseteq A^*$ be expressed in terms of their right quotients:

$$X = \bigcup_{a \in A} X_a \cdot a \cup \delta_X$$

$$Y = \bigcup_{a \in A} Y_a \cdot a \cup \delta_Y \quad .$$

Then

$$X \cup Y = \bigcup_{a \in A} (X_a \cup Y_a) \cdot a \cup (\delta_X \cup \delta_Y) \tag{11}$$

$$X \cap Y = \bigcup_{a \in A} (X_a \cap Y_a) \cdot a \cap (\delta_X \cap \delta_Y) \tag{12}$$

$$\overline{X} = \bigcup_{a \in A} \overline{X}_a \cdot a \cup (\lambda - \delta_X) \quad . \tag{13}$$

<u>Proof</u>: This is easily verified. □

Given a set of right language equations

$$X_i = \bigcup_{j=1}^{m} F_{i,j}(X) \cdot a_j \cup \delta_i \quad , \tag{14}$$

as in (9), we construct the set:

$$F_k = \bigcup_{j=1}^{m} \tilde{F}_{k,j}(X) \cdot a_j \cup \tilde{\delta}_k \quad , \tag{15}$$

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 in L_n write an expression involving $X_1, \ldots, X_n$,

$\cup$, $\cap$, $^{-}$, ϕ and I . Compute the functions $\tilde{F}_{k,j}$ for F_k , $j = 1,\ldots,m$, by using (11), (12) and (13). We call (15) the <u>right quotient equations generated</u> by (14).

In the example of Section 1 we have:

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

We first note that

$$\phi = \phi \cdot a \cup \phi \cdot b$$
$$I = I \cdot a \cup I \cdot b \cup \lambda \ .$$

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

$$\bar{X}_1 = X_2 \cdot a \cup \phi \cdot b \cup \lambda$$
$$\bar{X}_2 = \bar{X}_2 \cdot a \cup (\bar{X}_1 \cup X_2) \cdot b \quad .$$

Similarly:

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

etc. In this way we can construct a set of $2^{2^2} = 16$ equations of the form (15).

<u>Theorem 1</u> Any system of right language equations of the form (14) has a unique solution for each X_i , $i = 1,\ldots,n$. Furthermore each X_i is regular.

Proof: The system (15) of equations generated by (14) as described above has the form of quotient equations. Hence this system of equations can be solved using the fact that the equation

$$X = XB \cup C \ , \qquad \lambda \notin B \ ,$$

has the unique solution $X = CB^*$ which is regular if B and C are regular. 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 right language equations (14) are contained in the list of right quotient equations (15). Hence we have a unique solution for $X_1, \ldots, X_n$ which satisfies (14) since it is part of the solution of (15). $\square$

Returning to our example of Section 1, note that we are interested in finding $X_0 = X_1 \cap X_2$. In general, it is not necessary to find all 16 right quotient equations; we need only those functions that are "reachable" from X_0 , i.e. only the quotients of X_0 . This is in fact what we found as (3). Note also that the 16 distinct boolean functions of two variables in (15) do not necessarily define 16 distinct languages. In our example, $X_1 \cap \bar{X}_2 = \bar{X}_2$, though $y_1 \wedge y_2'$ and y_2' denote different boolean functions. However, the following system of equations generates 16 different languages:

$$\left.\begin{aligned} X_1 &= (\bar{X}_1 \cup \bar{X}_2)\cdot a \cup X_2\cdot b \\ X_2 &= \bar{X}_1\cdot a \cup ((\bar{X}_1 \cap \bar{X}_2) \cup (X_1 \cap X_2))\cdot b \end{aligned}\right\} \qquad (16)$$

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

In summary, we have shown in this section that (9) has a unique solution for $X_0 = G(X)$ and that X_0 is a regular language. We have yet to prove our claim that $X_0 = L(N)$.

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

$$f(y) = 1 \text{ iff } \lambda \in F(X) . \qquad (17)$$

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

<u>Basis, $r = 0$</u>

(a) If $f = 0$, then $F = \phi$ and (17) holds.

(b) If $f = 1$, then $F = I$ and (17) holds.

(c) If $f = y_i$ for some $i \in \{1,\dots,n\}$, then $F = X_i$. By assumption $y_i = 1$ iff $\lambda \in X_i$. Hence (17) holds.

<u>Induction Step, $r > 0$</u>

Assume now that (17) holds for g and G , as well as for h and H .

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

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

(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 $L(N)$ and X_0 agree as far as the empty word is concerned, for

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

By construction of (9), $y_i^o = 1$ iff $\lambda \in X_i$ for $i = 1,\ldots,n$. Thus Lemma 1 applies and $g(y^o) = 1$ iff $\lambda \in G(X)$. Next we would like to show that $w \in L(N)$ 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 (7), i.e.

$$y_i^\lambda = y_i^o$$

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

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

$$X/\lambda = X$$

and
$$X/a_j w = (X/w)/a_j \quad .$$

In view of these difficulties it is convenient to reverse the system (9) of equations. To simplify the notation we will let $V_{i,j} = X^{\rho}_{i,j}$, $V_i = X^{\rho}_i$, and $V = V_1, \ldots, V_n$. This formal reversal yields the system of left language equations:

$$\left.\begin{array}{l} V_i = a_1 \cdot F_{i,1}(V) \cup \ldots \cup a_m \cdot F_{i,m}(V) \cup \delta_i \; , \\ \qquad\qquad\qquad\qquad i = 1,\ldots,n \\ V_0 = G(V) \; . \end{array}\right\} \qquad (18)$$

Now we can deal with the left quotients of V .

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

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

<u>Proof</u>: 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. □

<u>Lemma 2</u> In the system (18) of left language equations, for all $i = 1,\ldots,n$, $a_j \in A$, $w \in A^*$

$$a_j w\backslash V_i = F_{i,j}(w\backslash V) \quad . \qquad (19)$$

Proof: We proceed by induction on $r = |w|$.

Basis, $r = 0$

Here $w = \lambda$, and (19) reduces to:

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

This follows immediately from (18).

Induction Step, $r > 0$

Assume (19) holds for wa_k , where $a_k \in A$. Then

$$\begin{aligned} a_j w a_k \backslash V_i &= a_k \backslash (a_j w \backslash V_i) \\ &= a_k \backslash F_{i,j}(w \backslash V) \\ &= F_{i,j}(wa_k \backslash V) \quad , \end{aligned}$$

by Proposition 2. □

Lemma 3 Let $w \in A^*$, let y^w be defined by (6) and (7) and let V be as in (18). Then for all $i = 1,\ldots,n$

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

Proof: We proceed by induction on $r = |w|$.

Basis, $r = 0$

Here $w = \lambda$, $y_i^\lambda = y_i^o$ and $\lambda^\rho \backslash V_i = V_i$. By construction of (9), $y_i^o = 1$ iff $\lambda \in V_i$.

Induction Step, $r > 0$

Assume the result holds for w and let $a_j \in A$. Then by (7)

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

By Lemma 2

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

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:

<u>Theorem 2</u> Let N be a network defined by (6) and let X_0 be the solution of the corresponding system of right language equations (9). Then $L(N) = X_0$.

<u>Proof</u>: By the definition of acceptance (8) we have

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

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

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

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

5. Boolean Automata

We now re-examine some correspondences between classes of automata and left language equations. Consider the following systems of equations:

$$\mathcal{E}_1 \qquad \begin{aligned} X_1 &= a\cdot X_1 \cup b\cdot X_2 \cup \lambda \\ X_2 &= a\cdot X_2 \cup b\cdot X_1 \end{aligned}$$

$$\mathcal{E}_2 \qquad \begin{aligned} X_1 &= a\cdot (X_1 \cup X_2) \cup b\cdot \phi \cup \lambda \\ X_2 &= a\cdot X_1 \cup b\cdot X_2 \end{aligned}$$

$$\mathcal{E}_3 \qquad \begin{aligned} X_1 &= a\cdot (\bar{X}_1 \cup \bar{X}_2) \cup b\cdot X_2 \cup \lambda \\ X_2 &= a\cdot \bar{X}_1 \cup b\cdot ((\bar{X}_1 \cap \bar{X}_2) \cup (X_1 \cap X_2)) \end{aligned}$$

The system $\mathcal{E}_3$ is most general, permitting all language functions in L_n to appear as coefficients of the letters. The system $\mathcal{E}_2$ uses only unions of the X_i ; note that empty unions are allowed. Finally the system $\mathcal{E}_1$ is most special, allowing only the X_i . It is well known that systems of type 1 correspond to deterministic automata and those of type 2 to nondeterministic automata. For the new systems of type 3 we can define a new 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.

A boolean automaton is a quintuple $\mathcal{B} = (A, Q, \tau, f^o, F)$, where

A is the input alphabet.

$Q = \{q_1, \ldots, q_n\}$ is the finite, nonempty <u>set of states</u>.

$\tau : Q \times A \to B_Q$ is the <u>transition function</u> which gives for each state and each letter a boolean function in B_Q .

f^o is the <u>initial function</u> in B_Q .

$F \subseteq Q$ is the set of <u>final states</u>.

For example let $\mathcal{B} = (\{a,b\}, \{q_1,q_2\}, \tau, q_1 \wedge q_2, \{q_2\})$, where τ is given in Figure 7.

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

Figure 7 A boolean automaton

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

$$\left.\begin{array}{ll}\text{(a)} & \tau(q_i, \lambda) = q_i \\ \text{(b)} & \tau(q_i, a_j w) = f_{i,j}(\tau(q_1, w), \ldots, \tau(q_n, w)) \text{ , where} \\ & f_{i,j} = \tau(q_i, a_j) \text{ .}\end{array}\right\} \quad (20)$$

Now for any $f \in B_Q$ define

$$\tau(f, w) = f(\tau(q_1, w), \ldots, \tau(q_n, w)) \text{ .} \qquad (21)$$

We remark, that we could replace (20)(b) by any of the following two definitions

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

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 $h = \tau(f^o, w)$. Then

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

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

To illustrate these concepts consider Figure 7. We find

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

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

The following theorem summarizes the main property of boolean automata.

Theorem 3

(a) For every boolean automaton $\mathcal{B} = (A, Q, \tau, f^o, F)$ 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 $L(\mathcal{B}) = [L(\mathcal{N})]^\rho$.

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

Proof:

(a) Let $\mathcal{B} = (A, Q, \tau, f^o, F)$ 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:

$$\left.\begin{array}{l} X_i = \bigcup_{a \in A} a \cdot F_{i,a} \cup \delta_i \quad , \quad i = 1, \ldots, n \\ X_0 = F^o \quad , \end{array}\right\} \qquad (22)$$

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^o corresponds to f^o in a similar way. Also $\delta_i = \lambda$ if $q_i \in F$ and $\delta_i = \phi$ otherwise.

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

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

or $w \in L(\mathcal{E})$ iff $\lambda \in F^o(w\backslash X_1, \ldots, w\backslash X_n)$. (23)

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

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

or $w \in L(\mathcal{B})$ iff $f^o(q_1^w(c), \ldots, q_n^w(c)) = 1$. (24)

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

$$\lambda \in w\backslash X_i \quad \text{iff} \quad q_i^w(c) = 1 \ . \qquad (25)$$

If we assume this claim, then Lemma 1 applies and $f^o(q_1^w(c), \ldots, q_n^w(c)) = 1$ iff $\lambda \in F^o(w\backslash X_1, \ldots, w\backslash X_n)$, thus showing that $L(\mathcal{B}) = L(\mathcal{E})$ by (23) and (24).

The proof of (25) 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 (25) holds.

Now we construct a network N so that its language will satisfy the reverse of (22). Namely, the network equations are of the form:

$$Y_i = (f_{i,1}(y) \wedge x_{a_1}) \vee \ \ldots \ \vee (f_{i,m}(y) \wedge x_{a_m}) , \ i = 1,\ldots,n$$
$$y^o = c$$
$$z = f^o(y) \quad .$$

It is then clear that the equations for $L(N)$ are obtained by reversing (22). Hence $[L(N)]^\rho = L(E) = L(B)$. Thus we have constructed N from B in such a way that $L(N) = [L(B)]^\rho$.

Reversing the argument proves the converse claim.

(b) Construct the derived deterministic automaton

$$A_B = (A, P, \mu, f^o, G)$$

as follows: $P = \{\tau(f^o, w) \mid w \in A^*\}$,

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

$c_i = 0$ otherwise} , and $\mu(\tau(f^o, w), a) = \tau(f^o, wa)$ for all $w \in A^*$ and $a \in A$. Note that $P \subseteq B_Q$ hence $\mathcal{A}_{\mathcal{B}}$ is a deterministic finite automaton. Furthermore one verifies that $L(\mathcal{A}_{\mathcal{B}}) = L(\mathcal{B})$. $\square$

In summary, boolean automata or the corresponding language equations provide a very 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 $\lceil \log_2 \log_2 n \rceil$.

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

$$X = a(X \cdot Y) \cup \lambda$$

$$Y = b \cdot Z$$

$$Z = \lambda$$

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

Bibliography

[1] Brzozowski, J.A., Canonical Regular Expressions and Minimal State Graphs for Definite Events, Mathematical Theory of Automata, New York, 1962, 529-561, Brooklyn, Polytechnic Institute of Brooklyn, 1963 (Symposia Series 12).

[2] Brzozowski, J.A., Regular Expressions from Sequential Circuits, IEEE Transactions, Vol. EC-13, No. 6, Dec. 1964, 741-744.

[3] Brzozowski, J.A. and Yoeli, M., Digital Networks, Englewood Cliffs, N. J., Prentice Hall, Inc., 1976.