# Linear regular languages. Part II
# The problem of synthesis

By G. T. HERMAN

## 1. Introduction

In Part I of this paper [3] we discussed the *problem of analysis* for linear sequential circuits, i.e. we gave an algorithm which for every language accepted by a linear sequential circuit (described by the circuit and the function mapping the symbols of the language into inputs to the circuit) produced a regular expression which described that language. We have also shown that the converse cannot be done, there is a regular expression such that the language described by it is not linear regular, i.e. its symbols cannot be mapped into inputs of a linear sequential circuit in such a way that the circuit will accept exactly those words which belong to the language. We shall assume that the reader is familiar with the terminology of [3].

The algorithm of [3] for the analysis of linear sequential circuits had the advantage over similar algorithms by its being a practical algorithm which can be implemented on a digital computer. Such implementation has been reported on in [4].

The problem of synthesis for linear sequential circuits is to give an algorithm which for any given regular expression decides whether or not it describes a linear regular language and, if that is indeed the case, the algorithm must provide us with a linear sequential circuit and a mapping of the symbols of the language into inputs of the circuit, such that the circuit will accept exactly those words which belong to the language. In this paper we shall describe an algorithm which will do this job. Unfortunately, from the practical point of view the algorithm will do little more than show that such algorithm exists, if implemented on a digital computer its operation would be so inefficient that it could not be applied even to very simple cases. This is a usual state of affairs with algorithms in automata theory, but it is our contention that this particular problem should have an implementable solution, similar in simplicity to the one for the problem of analysis. The reader should compare comments in § 5 of [3].

Similarly to [3], this paper will be introductory in the sense that it will make no assumption of knowledge on the part of the reader. Hence, some known definitions and results will be given and proved without reference to original sources.

## 2. Definitions

All definitions in [3] will be assumed to be known to the reader.

*Definition 1.* A *finite automaton* is a 5-tuple $M=\langle Q, \Sigma, q, \delta, F\rangle$, where
(i) $Q$ is a finite non-empty set of *states*,
(ii) $\Sigma$ is a finite non-empty set of *input symbols*,
(iii) $q\in Q$, the *initial state*,
(iv) $\delta: Q\times\Sigma \to Q$, the *direct transition function*,
(v) $F\subset Q$, the set of *accepting states*.

We extend the transition function to a mapping $\bar{\delta}$ from $Q\times I_\Sigma$ into $Q$ as follows:

$$\bar{\delta}(q, e)=q,$$

$$\bar{\delta}(q, xa)=\delta(\bar{\delta}(q, x), a)$$

for all $q\in Q$, $a\in\Sigma$ and $x\in I_\Sigma$.

Since, for all $q\in Q$ and $a\in\Sigma$, $\bar{\delta}(q, a)=\delta(q, a)$, we denote $\bar{\delta}$ by $\delta$ as well.

*Example 1.*

where $\langle\{q_0, q_1, q_2, q_3\}, \{a, b\}, q_0, \delta, \{q_2\}\rangle$

$$\delta(q_i, a)=q_i, \quad \text{for} \quad 0\leqq i\leqq 3,$$

$$\delta(q_i, b)=q_{i+1}, \quad \text{for} \quad 0\leqq i\leqq 2,$$

$$\delta(q_3, b)=q_3,$$

is a finite automaton.

*Example 2.*

$$\langle\{q_0, q_1\}, \{a, b\}, q_0, \delta, \{q_0\}\rangle$$

where $\delta(q_i, a)=q_i, \quad \text{for} \quad 0\leqq i\leqq 1,$

$$\delta(q_i, b)=q_{1-i}, \quad \text{for} \quad 0\leqq i\leqq 1,$$

is a finite automaton.

*Definition 2.* Let $M=\langle Q, \Sigma, q, \delta, F\rangle$ be a finite automaton and $x\in I_\Sigma$. We say that $M$ *accepts* $x$ if and only if $\delta(q, x)\in F$. Let $W\in L_\Sigma$. We say that $M$ *accepts* $W$ if and only if the set of those $x$ in $I_\Sigma$ which are accepted by $M$ is exactly $W$.

*Example 3.* The finite automaton given in Example 1 accepts the language described by

$$(((a^*b)(a^*b))a^*).$$

*Example 4.* The finite automaton given in Example 2 accepts the language described by

$$(a^*((ba^*)(ba^*))^*).$$

*Definition 3.* A finite automaton $M=\langle Q, \Sigma, q, \delta, F\rangle$ is said to be *linearly realizable* if and only if there exists a linear sequential circuit $C$ and functions $\alpha$ and $\varphi$ with the following properties. (We assume that the circuit $C$ has $k$ external input

wires and $n$ delays and can be described by matrices **A**, **B** and **C** as in Theorem 1 of [3].)

(i) $\alpha$ maps $\Sigma$ into $k$-tuples of 0's and 1's.

(ii) $\varphi$ maps $Q$ into $n$-tuples of 0's and 1's.

(iii) For each $p \in Q$ and $a \in \Sigma$,

$$\varphi(\delta(p, a)) = \varphi(p)\mathbf{A} \oplus \alpha(a)\mathbf{B}.$$

(iv) For each $p \in Q$,

$$p \in F \text{ if and only if } \varphi(p)\mathbf{C} = 1.$$

In such a case $C$ is said to be a *linear realization* of $M$.

*Example 5.* The finite automaton of Example 2 is linearly realizable. Its linear realization is given in Figure 3 of [3]. $\alpha$ is defined by

$$\alpha(a) = 0, \quad \alpha(b) = 1.$$

$\varphi$ is defined by

$$\varphi(q_0) = [1, 0], \quad \varphi(q_1) = [1, 1].$$

We shall see later on that the finite automaton of Example 1 is not linearly realizable. We note in passing that the definition of realization that is given here is somewhat restricted, but for the purpose of checking the linearity of regular languages it is as general as needed. For a discussion of various definitions of realization, see for instance [5].

*Definition 4.* An *initial subautomaton* of a finite automaton $M = \langle Q, \Sigma, q, \delta, F \rangle$ is the finite automaton $\langle Q', \Sigma, q, \delta', F' \rangle$, where

$$Q' = \{p | p \in Q \text{ and } p = \delta(q, x) \text{ for some } x \in I_\Sigma\},$$

$$\delta'(p, a) = \delta(p, a) \text{ for all } p \in Q' \text{ and } a \in \Sigma,$$

$$F' = F \cap Q'.$$

Intuitively, the initial subautomaton is that part of the automaton which consists of all the states which are accessible from the initial state.

*Example 6.* For the automata of Examples 1 and 2, the initial subautomaton is the automaton itself, since all states are accessible from the initial state.

The following basic result is easy to prove and we shall assume it in the rest of the paper without further reference to it.

*Proposition.* The language which is accepted by a finite automaton is the same as the language accepted by its initial subautomaton.

*Definition 5.* Let $C$ be a linear sequential circuit with $n$ delays. With each state $y = [y_1, \ldots, y_n]$ of $C$ we associate a mapping $\lambda_y^C$ from strings of inputs $x_1 \ldots x_t$ into $\{0, 1\}$ defined as follows:

$$\lambda_y^C(x_1 \ldots x_t) = y\mathbf{A}^t\mathbf{C} \oplus \sum_{i=1}^{t} x_i \mathbf{B}\mathbf{A}^{t-i}\mathbf{C}.$$

I.e. the value of $\lambda_y^C(x_1 \ldots x_t)$ is the same as the output would be at time $t+1$ if the machine was started in state $y$ and received the external input $x_i$ at time $i$.

*Example 7.* For the linear sequential machine $C$ of Figure 3 in [3] we have the following.

$$\lambda^C_{[1,0]}(x_1 \dots x_t) = \begin{cases} 1 \text{ if an even number of } x_i \text{ is } 1, \\ 0 \text{ otherwise,} \end{cases}$$

$$\lambda^C_{[1,1]}(x_1 \dots x_t) = \begin{cases} 0 \text{ if an even number of } x_i \text{ is } 1, \\ 1 \text{ otherwise,} \end{cases}$$

$$\lambda^C_{[0,1]}(x_1 \dots x_t) = \lambda^C_{[1,0]}(x_1 \dots x_t),$$

$$\lambda^C_{[0,0]}(x_1 \dots x_t) = \lambda^C_{[1,1]}(x_1 \dots x_t).$$

*Definition 6.* A linear sequential circuit $C$ is said to be *minimal* if and only if it does not have two different states $y_1$ and $y_2$ such that $\lambda^C_{y_1} = \lambda^C_{y_2}$.

*Example 8.* The linear sequential circuit of Figure 3 in [3] is not minimal.

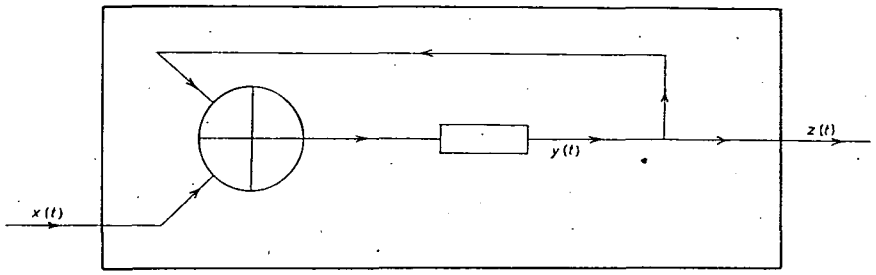


*Fig. 1*

*Example 9.* Let $C$ be the linear sequential circuit of Figure 1.

$$\lambda^C_1(x_1 \dots x_t) = \begin{cases} 1 \text{ if an even number of } x_i \text{ is } 1, \\ 0 \text{ otherwise,} \end{cases}$$

$$\lambda^C_0(x_1 \dots x_t) = \begin{cases} 0 \text{ if an even number of } x_i \text{ is } 1, \\ 1 \text{ otherwise.} \end{cases}$$

Hence $C$ is minimal.

*Definition 7.* Let $C$ and $C'$ be two linear sequential circuits. $C$ and $C'$ are said to be *equivalent* if and only if

$$\{\lambda^C_y | y \text{ is a state of } C\} = \{\lambda^{C'}_{y'} | y' \text{ is a state of } C'\}.$$

Intuitively speaking two linear sequential circuits are equivalent if and only if for every state $y$ of $C$ there is a state $y'$ of $C'$ such that $C$ started in state $y$ behaves the same way as $C'$ started in $y'$, and vice versa.

*Example 10.* The circuits in Examples 8 and 9 are equivalent.

## 3. Outline of the argument

Our aim is to provide an algorithm for the synthesis of linear sequential circuits. We are going to do this in the following way.

(a) We give an algorithm which for every regular expression produces a finite automaton such that the language described by the regular expression is the language accepted by the finite automaton.

(b) We prove that a language described by a regular expression is a linear regular language if and only if the initial subautomaton of the automaton produced by the algorithm in (a) is linearly realizable. In particular, we show that from the linear circuit which is the linear realization (if there is one) of the initial subautomaton, we can effectively produce a linear sequential circuit which accepts the language described by the regular expression.

(c) We give an algorithm which for any finite automaton decides whether or not its initial subautomaton is linearly realizable, and if it is, then the algorithm gives a linear sequential circuit which is a linear realization of it.

It is clear that in view of (b), the algorithms in (a) and (c) combine to give the algorithm required for the problem of synthesis for linear sequential circuits.

## 4. Synthesis for finite automata

*Theorem 1.* Let $\Sigma$ be any fixed finite, non-empty set. There is an algorithm which for any given regular expression $R$ over the alphabet $\Sigma$ will produce a finite automaton $M=\langle Q, \Sigma, q, \delta, F\rangle$ such that $M$ accepts the language $|R|$.

*Proof.* The algorithm builds up the finite automaton $M$ from finite automata which it has already produced for parts of $R$. We shall describe what it does for regular expressions of length 1, and then show how it produces the finite automaton for a regular expression of length greater than 1 from finite automata for regular expressions of shorter length.

If $R=o$, then $M=\langle\{q\}, \Sigma, q, \delta, \varnothing\rangle$,

where $\delta(q, a)=q$ for all $a\in\Sigma$.

If $R=e$, then $M=\langle\{q_1, q_2\}, \Sigma, q_1, \delta, \{q_1\}\rangle$,

where $\delta(q, a)=q_2$ for $q\in\{q_1, q_2\}$ and $a\in\Sigma$.

If $R=a$ where $a\in\Sigma$, then $M=\langle\{q_0, q_1, q_2\}, \Sigma, q_0, \delta, \{q_1\}\rangle$, where $\delta(q_0, a)=q_1$ and $\delta(q, b)=q_2$ otherwise.

We now show how to produce from a finite automaton $D=\langle\bar{Q}, \Sigma, \bar{q}, \bar{\delta}, \bar{F}\rangle$ which accepts a regular language $|P|$ and a finite automaton $E=\langle\bar{\bar{Q}}, \Sigma, \bar{\bar{q}}, \bar{\bar{\delta}}, \bar{\bar{F}}\rangle$ which accepts a regular language $|S|$ ($P$ and $S$ are regular expressions), the finite automata which accept the regular languages $|(P+S)|$ and $|(PS)|$, respectively.

Suppose $R$ is of the form $(P+S)$ and $D$ and $E$ are finite automata as defined above. The finite automaton $M=\langle Q, \Sigma, q, \delta, F\rangle$ which accepts $|R|$ is constructed as follows. $Q=\bar{Q}\times\bar{\bar{Q}}$, i.e. ordered pairs of elements from $\bar{Q}$ and $\bar{\bar{Q}}$. $q=\langle\bar{q}, \bar{\bar{q}}\rangle$.

For any $a\in\Sigma$, $p\in\bar{Q}$ and $r\in\bar{\bar{Q}}$,

$$\delta(\langle p, r\rangle, a) = \langle \bar{\delta}(p, a), \bar{\bar{\delta}}(r, a)\rangle,$$

$$F = \{\langle p, r\rangle | p\in\bar{F} \quad \text{or} \quad r\in\bar{\bar{F}}\}.$$

We leave it to the reader to show that $M$ accepts $|(P+S)|$.

Suppose $R$ is of the form $(PS)$ and $D=\langle \bar{Q}, \Sigma, \bar{q}, \bar{\delta}, \bar{F}\rangle$ and $E=\langle \bar{\bar{Q}}, \Sigma, \bar{\bar{q}}, \bar{\bar{\delta}}, \bar{\bar{F}}\rangle$ are finite automata which accept $|P|$ and $|S|$, respectively. The finite automaton $M=\langle Q, \Sigma, q, \delta, F\rangle$ which accepts $|R|$ is constructed as follows.

Let $Q'$ denote the set of all subsets of $\bar{\bar{Q}}$. $Q = \bar{Q}\times Q'$, i.e. a set of ordered pairs, where the first element is a state of $D$ and the second element is a subset of the states of $E$. $q=\langle \bar{q}, \varnothing\rangle$ if $\bar{q}\notin\bar{F}$ and $q=\langle \bar{q}, \{\bar{\bar{q}}\}\rangle$ if $\bar{q}\in\bar{F}$. $\delta$ is defined as follows. For any $p\in\bar{Q}$, $Q_1\subset\bar{\bar{Q}}$ and $a\in\Sigma$,

$$\delta(\langle p, Q_1\rangle, a) = \langle \bar{\delta}(p, a), Q_2\rangle,$$

where

$$Q_2 = \{r | r = \bar{\bar{\delta}}(s, a) \quad \text{for some} \quad s\in Q_1\}\cup Q_3,$$

where

$$Q_3 = \varnothing \quad \text{if} \quad \bar{\delta}(p, a)\in\bar{F} \quad \text{and} \quad Q_3 = \{\bar{\bar{q}}\} \quad \text{if} \quad \bar{\delta}(p, a)\in\bar{F}.$$

Finally,

$$F = \{\langle p, Q_1\rangle | p\in\bar{Q}, Q_1\subset\bar{\bar{Q}} \quad \text{and} \quad Q_1\cap\bar{\bar{F}}\neq\varnothing\}.$$

We now have to show that the finite automaton $F=\langle Q, \Sigma, q, \delta, F\rangle$ does indeed accept $|(PS)|$. The essence of the proof is the following. Given a word $w$ of $I_\Sigma$, there may be many ways of breaking up $w$ into two subwords. $w\in|(PS)|$ if and only if one of these ways is such that $w=w_1w_2$ and $w_1\in|P|$ and $w_2\in|S|$. $M$ keeps track simultaneously of all the possible states which $E$ might be in depending on the way $w$ has been broken up.

In order to complete this part of the proof, it is sufficient to prove the following claim.

For any word $w\in I_\Sigma$,

$$\delta(q, w) = \langle \bar{\bar{\delta}}(\bar{q}, w), Q_1\rangle,$$

where $r\in Q_1$ if and only if there exist $w_1$ and $w_2$ in $I_\Sigma$ such that $w=w_1w_2$, $w_1\in|P|$ and $\bar{\bar{\delta}}(\bar{\bar{q}}, w_2)=r$.

We leave the proof, which can best be done by induction on the length of $w$, to the reader.

If $R$ is of the form $P^*$ and $E=\langle \bar{Q}, \Sigma, \bar{q}, \bar{\delta}, \bar{F}\rangle$ is a finite automaton which accepts $|P|$, $M=\langle Q, \Sigma, q, \delta, F\rangle$ is constructed as follows.

Let $\bar{\bar{q}}$ be such that $\bar{\bar{q}}\notin\bar{Q}$. Let $\bar{\bar{Q}} = \bar{Q}\cup\{\bar{\bar{q}}\}$. Define $\bar{\bar{\delta}}: \bar{\bar{Q}}\times\Sigma \to \bar{Q}$ by

$$\bar{\bar{\delta}}(q, a) = \begin{cases} \bar{\delta}(q, a) & \text{if} \quad q\in\bar{Q} \quad \text{and} \quad a\in\Sigma, \\ \bar{\delta}(\bar{q}, a) & \text{if} \quad q=\bar{\bar{q}} \quad \text{and} \quad a\in\Sigma. \end{cases}$$

$Q$ is the set of all subsets of $\bar{\bar{Q}}$. $q=\{\bar{\bar{q}}\}$. For any $Q_1\subset\bar{\bar{Q}}$ and $a\in\Sigma$,

$$\delta(Q_1, a) = \{r | r = \bar{\bar{\delta}}(s, a) \quad \text{for some} \quad s\in Q_1\}\cup Q_2$$

where $Q_2=\{\bar{q}\}$ if $\bar{\bar{\delta}}(s, a)\in\bar{F}$ for some $s\in Q_1$ and $Q_2=\varnothing$ otherwise. Finally,

$$F=\{Q_1 | Q_1\subset\bar{\bar{Q}} \quad \text{and} \quad \bar{\bar{q}}\in Q_1\}.$$

The proof that $M$ accepts $|P^*|$ is very similar to the proof outlined above and we leave it to the reader.

Even though this theorem proves that the problem of synthesis is solved for finite automata, the algorithm described in it is such that the finite automata produced will in general be far from the simplest of the ones which do the job required. In fact, it will generally be so large that the implementation of the algorithm on an actual computing device is beyond the realm of practical possibility in all but the simplest cases. For instance, for the regular expression in Example 3 the finite automaton produced by the algorithm will have more than $4\times 10^{12}$ states, and the finite automaton for the regular expression in Example 4 will have over $10^{10^{10^{10}}}$ states. At the same time, both of the expressions denote languages that can be accepted by fairly simple machines (see Examples 1 and 2).

Although one could give algorithms which work faster than the one described above, there is no existing algorithm which works so fast as to be implementable for non-trivial regular expressions.

## 5. Some facts about linear sequential circuits and linear realizations

*Theorem 2.* Let $M=\langle Q, \Sigma, q, \delta, F\rangle$ be a finite automaton and $C$ be a linear sequential circuit which is a linear realization of $M$ with functions $\alpha$ and $\varphi$ (see Definition 3). Let $C$ be described by the matrices **A**, **B** and **C**. Then, for all $p\in Q$ and $x_1x_2\dots x_t$ in $I_\Sigma$,

$$\varphi\big(\delta(p, x_1x_2\dots x_t)\big)=\varphi(p)\mathbf{A}^t\oplus\sum_{i=1}^{t}\alpha(x_i)\mathbf{B}\mathbf{A}^{t-i}.$$

Furthermore,

$$\lambda^C_{\varphi(p)}\big(\alpha(x_1)\alpha(x_2)\dots\alpha(x_t)\big)=1 \text{ if and only if } \delta(p, x_1x_2\dots x_t)\in F.$$

*Proof* by induction on $t$.

If $t=0$,

$$\varphi\big(\delta(p, e)\big)=\varphi(p)=\varphi(p)\mathbf{A}^0.$$

Assume that the theorem is true for all words of length $t$. Then

$$\varphi\big(\delta(p, x_1x_2\dots x_tx_{t+1})=\varphi\big(\delta(\delta(p, x_1x_2\dots x_t), x_{t+1})\big)=$$

$$=\varphi\big(\delta(p, x_1x_2\dots x_t)\big)\mathbf{A}\oplus\alpha(x_{t+1})\mathbf{B}=\left(\varphi(p)\mathbf{A}^t\oplus\sum_{i=1}^{t}\alpha(x_i)\mathbf{B}\mathbf{A}^{t-i}\right)\mathbf{A}\oplus\alpha(x_{t+1})\mathbf{B}=$$

$$=\varphi(p)\mathbf{A}^{t+1}\oplus\sum_{i=1}^{t+1}\alpha(x_i)\mathbf{B}\mathbf{A}^{t+1-i}.$$

The second part of the theorem follows directly from Definitions 3 and 5.

*Theorem 3.* Let $C$ be a linear sequential circuit with $n$ delays. Let $y_1$ and $y_2$ be any two states of $C$. If there exists a sequence $x_1, x_2, \ldots, x_{n-1}$ of external input conditions such that

$$\lambda^C_{y_1}(x_1 x_2 \ldots x_t) = \lambda^C_{y_2}(x_1 x_2 \ldots x_t).$$

for all $t$, $0 \leqq t \leqq n-1$, then

$$\lambda^C_{y_1} = \lambda^C_{y_2}.$$

(This is sometimes expressed by saying that $C$ satisfies the $n$-diagnosability condition.)

*Proof.* Assume that $C$ can be described by matrices **A**, **B** and **C**. If $\lambda^C_{y_1}(x_1 x_2 \ldots x_t) = \lambda^C_{y_2}(x_1 x_2 \ldots x_t)$, then

$$y_1 \mathbf{A}^t \mathbf{C} \oplus \sum_{i=1}^{t} x_i \mathbf{B} \mathbf{A}^{t-i} \mathbf{C} = y_2 \mathbf{A}^t \mathbf{C} \oplus \sum_{i=1}^{t} x_i \mathbf{B} \mathbf{A}^{t-i} \mathbf{C}$$

and so

$$y_1 \mathbf{A}^t \mathbf{C} = y_2 \mathbf{A}^t \mathbf{C}. \tag{1}$$

Since this is true for $0 \leqq t \leqq n-1$, it is also true for $t \geqq n$. This follows from the fact (Cayley—Hamilton theorem) that any power of **A** can be expressed as a linear combination of powers of **A** less than $n$. Thus we have that (1) holds for every $t$. But then for any sequence $x_1 x_2 \ldots x_t$,

$$\lambda^C_{y_1}(x_1 x_2 \ldots x_t) = \lambda^C_{y_2}(x_1 x_2 \ldots x_t)$$

and so $\lambda^C_{y_1} = \lambda^C_{y_2}$.

*Theorem 4.* For every linear sequential circuit $C$ there exists a minimal linear sequential circuit $C'$ which is equivalent to $C$.

*Proof.* Assume that $C$ is described by the matrices **A**, **B** and **C**. (**A** is $n \times n$, **B** is $k \times n$ and **C** is $n \times 1$.) Let **K** be the $n \times n$ matrix

$$[\mathbf{C}, \mathbf{AC}, \ldots, \mathbf{A}^{n-1}\mathbf{C}].$$

First we prove that $C$ is minimal if and only if **K** is a non-singular matrix.

$C$ is not minimal

if and only if

there exist $y_1$ and $y_2$ such that $\lambda^C_{y_1} = \lambda^C_{y_2}$ and $y_1 \neq y_2$,

if and only if

there exist $y_1$ and $y_2$ such that $y_1 \neq y_2$ and

$$y_1 \mathbf{A}^t \mathbf{C} = y_2 \mathbf{A}^t \mathbf{C} \quad \text{for} \quad 0 \leqq t \leqq n-1,$$

if and only if

there exist $y_1$ and $y_2$ such that $y_1 \neq y_2$ and

$$y_1 \mathbf{K} = y_2 \mathbf{K}$$

if and only if

**K** is singular.

So if **K** is non-singular, then $C$ is already minimal and there is nothing to prove. Let us therefore assume that **K** is singular.

Now we pick a row of **K** which is the linear combination of previous rows. Since the underlying field is the field of two elements this means that the particular row of **K**, say the $j$'th, is the modulo 2 sum of some of the earlier rows. What is the physical significance of the $j$'th row of $K$ being the sum of previous rows? It is that the $j$'th delay in the circuit is superfluous, since its behaviour can be obtained from the output of the other delays.

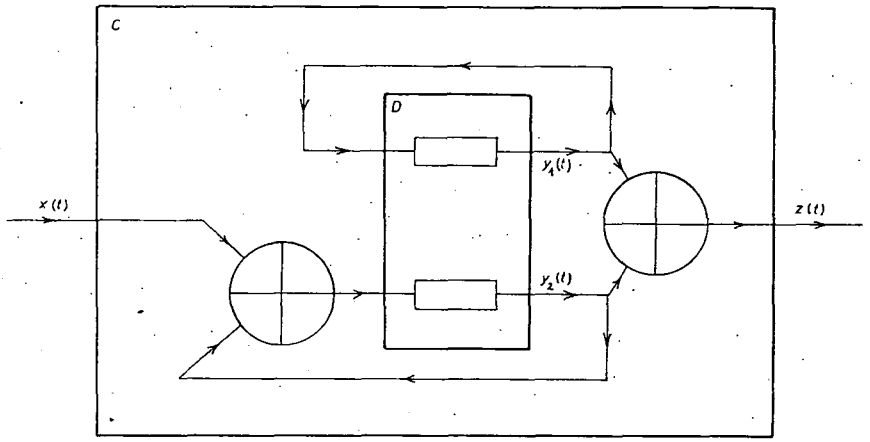


*Fig. 2*

To see this consider the following argument. We separate the delays of a linear sequential circuit $C$ by considering them to form a separate circuit $D$ with $n$ external input wires and $n$ external output wires. For instance for the circuit of Figure 3 in [3], this would give us Figure 2. For $1 \leqq j \leqq n$, the $j$'th external output of $D$ at time $t+1$ is the same as the $j$'th external input of $D$ at time $t$. Another way of looking at this, is that $D$ takes a state of $C$ as an input and returns the same state of $C$ as an output one unit of time later.

Now suppose that we have a linear sequential circuit $C'$ which is the same as $C$, except that $D$ is replaced by a circuit $D'$ with $n$ external inputs and $n$ external outputs, such that if $D'$ is given a state $y_1$ of $C$ as an input, then it returns one unit of time later state $y_2$ of $C$ such that

$$\lambda^C_{y_1} = \lambda^C_{y_2}.$$

Clearly, such a $C'$ is equivalent to $C$.

We are now going to show that if **K** is singular, we can always find such a $D'$ with only $n-1$ delays. One of the many possible ways of constructing such a $D'$ is the following.

Suppose the $j$'th row is the first row of **K** which is the sum of some of the previous rows. Let us denote the rows of **K** by $\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_n$ and let $S$ be that subset of $\{1, \ldots, j-1\}$ such that

$$\mathbf{K}_j = \sum_{i \in S} \mathbf{K}_i.$$

If $j \leqq i \leqq n-1$, the input wire of the $i$'th delay of $D'$ is connected to the $(i+1)$'st external input wire to $D'$, and the output wire of the $i$'th delay of $D'$ is connected to the $(i+1)$'st external output wire of $D'$. If $1 \leqq i \leqq j-1$ the output wire of the $i$'th delay of $D'$ is connected to the $i$'th external output wire of $D'$. If $i \notin S$, the input wire of the $i$'th delay of $D'$ is connected to the $i$'th external input wire of $D'$. If $i \in S$, the input wire of the $i$'th delay of $D'$ is connected to the output wire of an exclusive or gate whose input wires are the $i$'th and $j$'th external input wires to $D'$. Note that the $j$'th external output wire of $D'$ is not connected to anything and so it never carries a pulse (it is earthed).

If the input to $D'$ is the state $y_1 = [a_1, a_2, \ldots, a_n]$, then the output to $D'$ one unit of time later will be

$$y_2 = [a_1 + b_1, a_2 + b_2, \ldots, a_{j-1} + b_{j-1}, 0, a_{j+1}, \ldots, a_n],$$

where

$$b_i = \begin{cases} 0 & \text{if } i \notin S, \\ a_j & \text{if } i \in S. \end{cases}$$

In order to show that $\lambda^C_{y_1} = \lambda^C_{y_2}$, it is sufficient to prove that $y_1\mathbf{K} = y_2\mathbf{K}$ (see beginning of this proof).

$$y_2\mathbf{K} = [a_1 + b_1, a_2 + b_2, \ldots, a_{j-1} + b_{j-1}, 0, a_{j+1}, \ldots, a_n] \begin{bmatrix} \mathbf{K}_1 \\ \mathbf{K}_2 \\ \vdots \\ \mathbf{K}_n \end{bmatrix} =$$

$$= \sum_{i=1}^{j-1} a_i\mathbf{K}_i + \sum_{i=1}^{j-1} b_i\mathbf{K}_i + \sum_{i=j+1}^{n} a_i\mathbf{K}_i =$$

$$= \sum_{i=1}^{j-1} a_i\mathbf{K}_i + \sum_{i \in S} a_j\mathbf{K}_i + \sum_{i=j+1}^{n} a_i\mathbf{K}_i =$$

$$= \sum_{i=1}^{j-1} a_i\mathbf{K}_i + a_j\mathbf{K}_j + \sum_{i=j+1}^{n} a_i\mathbf{K}_i =$$

$$= \sum_{i=1}^{n} a_i\mathbf{K}_i =$$

$$= y_1\mathbf{K}.$$

We have now obtained a circuit $C'$ which is equivalent to $C$ and has fewer delays than $C$. If this circuit is not minimal, then we can repeat this process. Since $C$ has only a finite number of delays, sooner or later we must find a minimal $C'$ which is equivalent to $C$.

*Example 11.* Consider the circuit $C$ of Figure 2. For this circuit (see Example 5 in [3]).

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

$$\mathbf{K} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{K}_1 = \mathbf{K}_2 = [1, 1].$$

So in the method described above $j=2$ and $S=\{1\}$. This gives us the $D'$ and $C'$ shown in Figure 3.

It is easy to see that this circuit is equivalent to the one in Figure 1 ($y_2(t)$ never carries a pulse).

We point out by the way, that the proof of Theorem 4 is constructive. Given a linear sequential circuit, we can actually construct a minimal linear sequential circuit equivalent to it using the proof of Theorem 4.

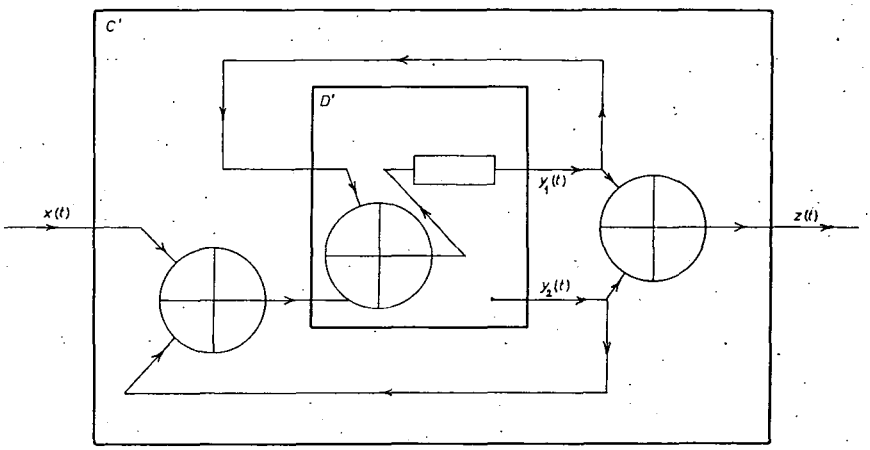


*Fig. 3*

## 6. The equivalence of linear realizability of finite automata and linearity of regular expressions

*Theorem 5.* Let $R$ be a regular expression and $M$ be a finite automaton which accepts $|R|$. Then $|R|$ is a linear regular language if and only if the initial subautomaton of $M$ is linearly realizable. Furthermore, if the initial subautomaton of $M$ is linearly realizable by a linear sequential circuit $C$, then from $C$ and the mappings $\varphi$ and $\alpha$ we can effectively produce a linear sequential circuit $C'$ and a function $f$ such that $C'$ accepts the language $|R|$ using $f$.

*Proof.* First suppose that the initial subautomaton of $M=\langle Q, \Sigma, q, \delta, F\rangle$ is linearly realizable by a linear sequential circuit $C$. Let $\alpha$ and $\varphi$ denote the mappings described in Definition 3.

We now construct a circuit $C'$ which accepts the language $|R|$ using $\alpha$ as the $f$ of Definition 9 in [3].

Consider $\varphi(q)=[y_1, \ldots, y_n]$. We consider each delay of $C$ in turn. If $y_i=0$, we make no alteration to the $i$'th delay of $C$. If $y_i=1$, then we connect the output wire of the $i$'th delay of $C$ to one of the input wires of an exclusive or gate (newly introduced for this purpose) and we connect the output wire of the exclusive or gate to all the wires to which the output wire of the delay used to be connected (all

the old connections being removed). The other input wire of the exclusive or gate is not yet connected up. When we changed all delays in this way, we introduce one more delay (to be considered the first delay of $C'$) whose input wire is not connected to anything (i.e., it is earthed) and whose output wire is connected to the so far free input wires of the newly introduced exclusive or gates. No other alteration is done to $C$ to obtain $C'$.

For example if $C$ is as in Figure 3 of [3], then $C'$ is given in Figure 4.

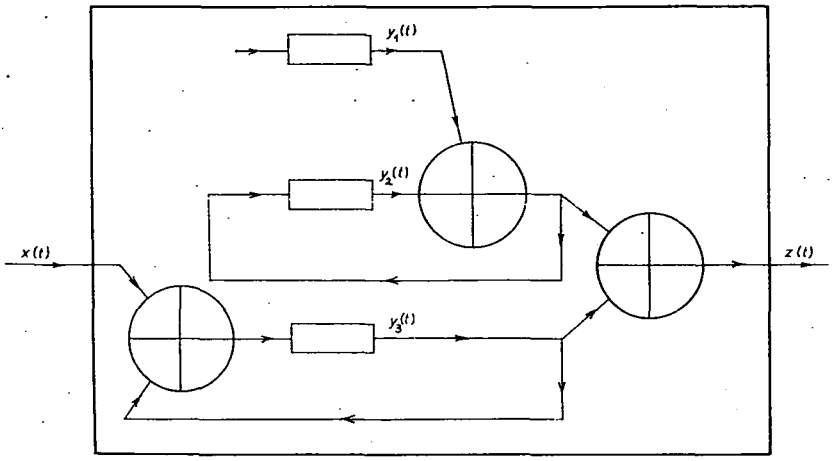


*Fig. 4*

After time $t=1$, the first delay of $C'$ makes no contribution to the behaviour of $C'$, hence we have that

$$\lambda^{C'}_{[1,0,\ldots,0]} = \lambda^{C}_{\varphi(q)}.$$

This together with Theorem 2 shows that if we let the function $f$ of Definition 9 in [3] to be $\alpha$, then a word $w$ in $I_\Sigma$ is accepted by $M$ if and only if it is accepted by $C'$. In particular, we have shown that $|R|$ is a linear regular language.

Conversely, let us assume that $|R|$ is a linear regular language and let $C$ be a linear sequential circuit which accepts $|R|$. Let $C'$ be a minimal linear sequential circuit which is equivalent to $C$. We shall prove that $C'$ is a linear realization of the initial subautomaton of $M$.

Since the number of external input wires for $C$ and $C'$ are the same, we can take $\alpha$ of Definition 3 to be the $f$ of Definition 9 in [3]. $\varphi$ is defined as follows.

For every state $y$ of $C$ there is one state $y'$ of $C'$ such that

$$\lambda^{C}_{y} = \lambda^{C'}_{y'}$$

(otherwise $C$ and $C'$ would not be equivalent). Furthermore, there is only one such $y'$, for otherwise $C'$ would not be minimal. Let $\mu(y)$ denote this unique $y'$.

Given a state $p$ of the initial subautomaton of $M$, there exists an $x = x_1 x_2 \ldots x_t$ in $I_\Sigma$ such that

$$p = \delta(q, x_1 x_2 \ldots x_t).$$

We define (using the matrices $\mathbf{A}'$, $\mathbf{B}'$ and $\mathbf{C}'$ which describe $C'$)

$$\varphi(p) = \mu[1, 0, \ldots, 0]\mathbf{A}'^{t} \oplus \sum_{i=1}^{t} \alpha(x_i)\mathbf{B}'\mathbf{A}'^{t-i}.$$

The difficulty with this definition is that it is not necessarily unique. It is possible that for some $\bar{x} = \bar{x}_1\bar{x}_2 \ldots \bar{x}_{\bar{t}}$ in $I_\Sigma$, $x \neq \bar{x}$ and yet $\delta(q, x) = \delta(q, \bar{x})$. We must show that in such cases

$$(2) \qquad \mu[1, 0, \ldots, 0]\mathbf{A}'^{t} \oplus \sum_{i=1}^{t} \alpha(x_i)\mathbf{B}'\mathbf{A}'^{t-i} = \mu[1, 0, \ldots, 0]\mathbf{A}'^{\bar{t}} \oplus \sum_{i=1}^{\bar{t}} \alpha(\bar{x}_i)\mathbf{B}'\mathbf{A}'^{\bar{t}-i}.$$

Let us denote the left hand side of (2), which is a state of $C'$, by $y_1$ and the right hand side of (2) by $y_2$. Since $C'$ is a minimal linear sequential circuit, to show the equality, it is sufficient to prove that

$$\lambda^{C'}_{y_1} = \lambda^{C'}_{y_2}.$$

We are now going to show that for any sequence $s_1 s_2 \ldots s_j \in I_\Sigma$

$$\lambda^{C'}_{y_1}\big(\alpha(s_1)\alpha(s_2) \ldots \alpha(s_j)\big) = \lambda^{C'}_{y_2}\big(\alpha(s_1)\alpha(s_2) \ldots \alpha(s_j)\big).$$

In view of Theorem 3, this is sufficient.

$$\lambda^{C'}_{y_1}\big(\alpha(s_1) \ldots \alpha(s_j)\big) =$$

$$= y_1\mathbf{A}'^{j}\mathbf{C}' \oplus \sum_{i=1}^{j} \alpha(s_i)\mathbf{B}'\mathbf{A}'^{j-i}\mathbf{C}' =$$

$$= \mu[1, 0, \ldots, 0]\mathbf{A}'^{t+j}\mathbf{C}' \oplus \sum_{i=1}^{t} \alpha(x_i)\mathbf{B}'\mathbf{A}^{t+j-i}\mathbf{C}' \oplus \sum_{i=1}^{j} \alpha(s_i)\mathbf{B}'\mathbf{A}'^{j-1}\mathbf{C}' =$$

$$= \lambda^{C'}_{\mu[1,0,\ldots,0]}\big(\alpha(x_1) \ldots \alpha(x_t)\alpha(s_1) \ldots \alpha(s_j)\big) =$$

$$= \lambda^{C}_{[1,0,\ldots,0]}\big(\alpha(x_1) \ldots \alpha(x_t)\alpha(s_1) \ldots \alpha(s_j)\big) =$$

$$= \begin{cases} 1 \text{ if } x_1 \ldots x_t s_1 \ldots s_j \in |R|, \\ 0 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 1 \text{ if } \delta(q, x_1 \ldots x_t s_1 \ldots s_j) \in F, \\ 0 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 1 \text{ if } \delta\big(\delta(q, x_1 \ldots x_t), s_1 \ldots s_j\big) \in F, \\ 0 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 1 \text{ if } \delta(p, s_1 \ldots s_j) \in F, \\ 0 \text{ otherwise} \end{cases}$$

$$= \begin{cases} 1 \text{ if } \delta\big(\delta(q, \bar{x}_1 \ldots \bar{x}_{\bar{t}}), s_1 \ldots s_j\big) \in F, \\ 0 \text{ otherwise} \end{cases}$$

$$= \lambda^{C'}_{y_2}\big(\alpha(s_1) \ldots \alpha(s_j)\big).$$

All that is left to show is that conditions (iii) and (iv) in Definition 3 are satisfied.

For any $p$ in the initial subautomaton of $M$ and for any $a$ in $\Sigma$, $p=\delta(q, x)$ for some $x=x_1 \ldots x_t$ in $I_\Sigma$ and

$$\begin{aligned}\varphi(\delta(p, a)) &= \varphi(\delta(\delta(q, x), a)) = \\ &= \varphi(\delta(q, xa)) = \\ &= \mu[1, 0, \ldots, 0]\mathbf{A}'^{t+1} \oplus \sum_{i=1}^{t} \alpha(x_i)\mathbf{B}'\mathbf{A}'^{t+1-i} \oplus \alpha(a)\mathbf{B}' = \\ &= \left(\mu[1, 0, \ldots, 0]\mathbf{A}'^{t} \oplus \sum_{i=1}^{t} \alpha(x_i)\mathbf{B}'\mathbf{A}'^{t-i}\right)\mathbf{A}' \oplus \alpha(a)\mathbf{B}' = \\ &= \varphi(\delta(q, x))\mathbf{A}' \oplus \alpha(a)\mathbf{B}' = \\ &= \varphi(p)\mathbf{A}' \oplus \alpha(a)\mathbf{B}'.\end{aligned}$$

$$\varphi(p)\mathbf{C}' = 1$$

if and only if

$$\left(\overset{\circ}{\mu}[1, 0, \ldots, 0]\mathbf{A}'^{t} \oplus \sum_{i=1}^{t} \alpha(x_i)\mathbf{B}'\mathbf{A}'^{t-1}\right)\mathbf{C}' = 1$$

if and only if

$$C \text{ accepts } x$$

if and only if

$$x \in |R|$$

if and only if

$$M \text{ accepts } x$$

if and only if

$$\delta(q, x) \in F$$

if and only if

$$p \in F.$$

*Corollary.* There are finite automata which are not linearly realizable.

*Proof.* The automaton of Example 1 is such. This is because this automaton accepts the language described in Example 3, and this language is not a linear regular language. (See Theorem 3 of [3].)

## 7. The linear realizability of finite automata

*Theorem* 6. If a linearly realizable finite automaton $M=\langle Q, \Sigma, q, \delta, F\rangle$ has $n'$ states and $k'$ symbols in $\Sigma$, then it is linearly realizable by a linear sequential circuit with at most $k'$ external input wires and at most $n'$ delays.

*Proof.* Suppose $M$ has a linear realization $C$ with $k$ external input wires and $n$ delays, where $k>k'$ or $n>n'$ or both. Suppose $\alpha$ and $\varphi$ are the mappings as in Definition 3.

First we deal with the case when $n>n'$.

Let $q_1, \ldots, q_{n'}$ be the $n'$ states of $M$. Consider the $n' \times n$ matrix **F** whose $i$'th row is $\varphi(q_i)$. At least $n-n'$ columns of this matrix **F** will be linearly dependent on the other $n'$ columns, and by relabeling the delays we may assume that it is the last $n-n'$ columns.

Now we proceed as in the proof of Theorem 4. We consider the subcircuit $D$ with $n$ external inputs and $n$ external outputs which contains all the delays and nothing else. We replace this by a circuit $D'$ with $n'$ delays. For $1 \leqq i \leqq n'$, the input wire to the $i$'th delay is connected to the $i$'th external input wire to $D'$, and the output wire of the $i$'th delay is connected to the $i$'th external output wire of $D'$.

The other external input wires are not connected to anything. For $n' < i \leqq n$, the $i$'th external output wire of the $D'$ is connected to the delays through exclusive or gates in such a way that its output is the modulo 2 sum of the outputs of those delays which correspond to those of the first $n'$ columns of **F** which added together give the $i$'th column.

The circuit $C'$ which we get by replacing $D$ in $C$ by $D'$ is clearly equivalent to $C$ and is also a linear realization of $M$ with functions $\alpha' = \alpha$ and $\varphi'$ such that $\varphi'(p)$ is the vector consisting of the first $n'$ elements of $\varphi(p)$ (after relabeling of the delays of $C$).

The case when $k > k'$ can be similarly taken care of. $C$ is altered by attaching in front of it a circuit consisting of exclusive or gates only (no delays) which has $k'$ external input wires and $k$ external output wires, the latter being attached to the external input wires of $C$. The exact nature of this additional circuit is determined by the linear dependencies between the columns of the matrix whose $i$'th row is $\alpha(s_i)$, where $s_i$ is the $i$'th element of $\Sigma$.

*Theorem 7.* There is an algorithm which for any finite automaton decides whether or not its initial subautomaton is linearly realizable, and, in case it is, the algorithm gives a linear sequential circuit which is a linear realization of it.

*Proof.* First of all, it should be obvious that there is an algorithm which from a given automaton produces its initial subautomaton. Let us assume that this initial subautomaton has $n$ states and $k$ symbols in its alphabet. If it is linearly realizable, then it has a linear realization with $n$ delays and $k$ external input wires. (At most $n$ or $k$, by Theorem 6, if less, then additional delays and external input wires can be introduced and earthed.) This linear realization can be described by an $n \times n$ matrix **A**, a $k \times n$ matrix **B** and a $n \times 1$ matrix **C** of 0's and 1's. Furthermore given three such matrices we can easily produce a linear sequential circuit which is described by them, and any two circuits described by them will be equivalent and be linear realizations of the same finite automata. (Only the matrices enter the definition of equivalence and linear realization.) Also, given matrices **A**, **B** and **C** it is easy to check (using Definition 3) whether or not the circuit described by them is a linear realization of a given automaton.

So our algorithm will look like this. Try all possible $n \times n$ matrices **A**, $k \times k$ matrices **B** and $n \times 1$ matrices **C** (there will be $2^{n(n+k+1)}$ possibilities). Check one by one whether the circuits described by them are linear realizations of the initial subautomaton of the given automaton. If we find such **A**, **B** and **C**, then our work is done, if we exhaust all possibilities without finding them, then the initial subautomaton is not linearly realizable.

There are more efficient algorithms than the one described above to do what is required in Theorem 7. (One such could be based on the method of Cohn & Even [1].) However, in view of the comments after Theorem 1, it is clear that our total algorithm for the synthesis of linear sequential circuits cannot be made practicable and so we sacrificed efficiency for ease of proof in Theorem 7.

## 8. Conclusions

The algorithms described in Theorems 1, 5 and 7 together provide us with an algorithm for the synthesis of linear sequential circuits. However, this is a very roundabout as well as inefficient way of doing things, and the possibility of a direct synthesis from regular expressions to circuits remains an intriguing open problem.

In this direction, the reader may find useful two books related to linear sequential circuits which appeared since the writing of Part I. These are [2] and [6].

DEPT. OF COMPUTER SCIENCE
STATE UNIVERSITY OF NEW YORK
AT BUFFALO

## 9. References

[1] COHN, M. & S. EVEN, Identification and minimization of linear machines, *IEEE Trans.*, v. EC-14, 1965, pp. 367—376.
[2] HARRISON, M. A., *Lectures on Linear Sequential Machines*, Academic Press, New York, 1970.
[3] HERMAN, G. T., Linear regular languages, Part I, *Acta Cybernetica*, v. 1, 1969, pp. 3—11.
[4] HERMAN, G. T., Analysis of linear sequential circuits, *Proceedings of the Fourth Annual Princeton Conference on Information Sciences and Systems*, 1970, pp. 392—396.
[5] HERMAN, G. T., When is a sequential machine the realization of another? *Mathematical Systems Theory*, v. 5, 1971, pp. 115—127.
[6] REUSCH, B., *Lineare Automaten*, Bibliographisches Institut, Mannheim, 1969.

*(Received June 24, 1970)*