INFORMATION AND CONTROL 5, 204–212 (1962)

Sequence Generators, Graphs, and Formal Languages*†

ARTHUR W. BURKS

Department of Philosophy, University of Michigan, Ann Arbor, Michigan

AND

JESSE B. WRIGHT

Logic of Computers Group, University of Michigan, Ann Arbor, Michigan

A sequence generator is a finite graph, more general than, but akin to, the usual state diagram associated with a finite automaton. The nodes of a sequence generator represent complete states, and each node is labeled with an input and an output state. An element of the behavior of a sequence generator is obtained by taking the input and output states along an infinite path of the graph.

Sequence generators may be associated with formulas of the monadic predicate calculus, in which the individual variables range over the times 0, 1, 2, 3, $\cdots$, and the predicate variables represent complete states, input states, and output states. An unrestricted singulary recursion is a formula in which the complete state at time $\tau + 1$ is expressed as a truth-function of the complete state at time τ and the input states from times $\tau + 1$ to $\tau + h$. Necessary and sufficient conditions are given for a formula derived from a sequence generator being equivalent to an unrestricted singulary recursion.

The fundamental concept is that of a sequence generator. A *sequence generator* is a *finite*, *directed*, *labeled* graph. Each node may or may not be labeled as a *root R*. Each node is labeled with a pair of truth values ("*t*" for true, "*f*" for false). Figure 1 is an example.

We are particularly interested in what we call the "behavior" of a sequence generator (see Fig. 2). The behavior of a sequence generator

* This research was supported by the Office of Naval Research Contract No. Nonr-1224(21).

† This paper is a continuation of Burks and Wright (1962). In the first part of the present paper we relate the sequence generators of Burks and Wright (1962) to graphs; the first definability theorem below is a graph-theoretic version of one of the main results of Burks and Wright (1962). The second definability theorem below is new.

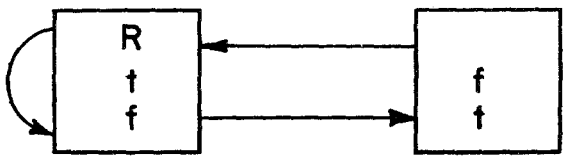

FIG. 1. A sequence generator

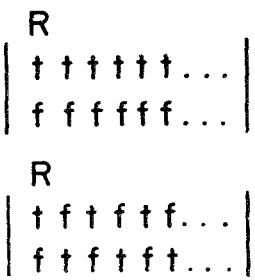

FIG. 2. Two elements of the behavior of Fig. 1

is a set of matrices of truth values. The pairs of truth values taken from the nodes along an infinite path of the graph constitute a two-by-omega matrix. Such a two-by-omega matrix belongs to the *behavior* of a sequence generator if and only if it is taken from a path of the graph which begins with a root. For example, the first element of Fig. 2 is taken from the path of Fig. 1 that starts with the root and always stays in the root, while the second element is taken from the path that starts with the root and oscillates back and forth between it and the other node. This concept of behavior may be connected to the ordinary one by calling the upper truth value of a node an *input state* and calling the lower truth value of a node an *output state*. A sequence generator then determines a relation between infinite input sequences and infinite output sequences, which relation constitutes its behavior.

The inputs and outputs labeling nodes may be vectors of truth values rather than single truth values. Our results all apply to this more general case, but in the interest of simplicity we will present them in terms of the case where the input and output vectors are of length one.

Digital computers or finite automata are closely related to a special type of sequence generator called a *deterministic* sequence generator. Figure 3 is deterministic. A sequence generator is deterministic if it satisfies these two conditions: (a) There is exactly one root whose input

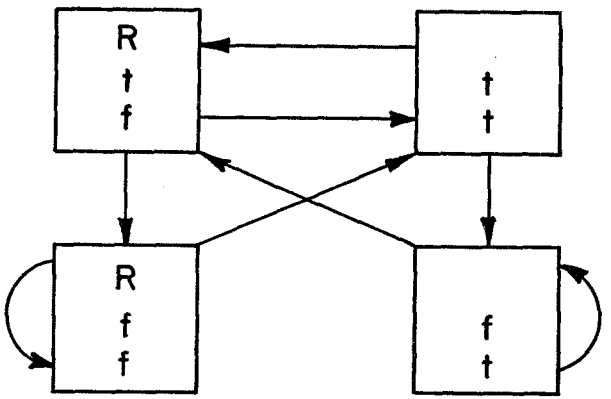

FIG. 3. Deterministic sequence generator for a binary counter

truth value is t, and exactly one root whose input value is f. (b) For each node there is exactly one node immediately following the given node and having t as its input, and exactly one node immediately following the given node and having f as its input. The deterministic sequence generator of Fig. 3 corresponds, in fact, to a binary counter.

Consider a graph derived from a digital computer or finite automaton in the following way. Each pair consisting of an input state and an internal state is a complete state. A node of the graph is provided for each complete state. The labels on a node give the input state and output state associated with the corresponding complete state of the automaton. Directed arrows of the graph indicate direct transitions between complete states. Any graph derived from a digital computer in this way is a deterministic sequence generator, and each deterministic sequence generator corresponds to a digital computer.

The rows of truth values in the infinite matrices introduced earlier define predicates on the natural numbers. This suggests the use of logical formulas to represent sequence generators. We do this in an interpreted system called the sequential calculus (see Büchi (1962)). The symbols of the sequential calculus are: individual variables, interpreted as ranging over the natural numbers; monadic predicate variables, interpreted as ranging over predicates of natural numbers; the individual constant zero; the successor function; truth-functional connectives; and quantifiers on both individual and predicate variables.

It is possible to associate with each sequence generator a formula of the sequential calculus. Let Γ be any sequence generator with no more than

2^n nodes. In addition to the labels in Γ assign to each node a distinct n-tuple of truth values, called the code-word of that node. Predicates $s_1 \cdots s_n$ (abbreviated s) are associated with a sequence generator in such a way that for a node-path $x_0, x_1, x_2, \cdots$, with x_0 as a root, $s_i(\tau)$ is true if and only if the ith truth value assigned to x_τ is t. The sequence generator Γ, without reference to its behavior, is represented by the conjunction of the formulas for $s_1(0) \cdots s_n(0)$, $s_1(\tau') \cdots s_n(\tau')$. The behavior of Γ is represented by conjoining equivalences for the input predicate i and for the output predicate o, and existentially quantifying s. The formula corresponding to Fig. 1 is

$$(\exists s)\{s(0) \,\&\, (\forall \tau)[(s(\tau) \vee s(\tau')) \,\&\, (i(\tau) \equiv s(\tau)) \,\&\, (o(\tau) \equiv \overline{s(\tau)})]\}. \quad (1)$$

The existential predicate quantifier "there is an s" asserts the existence of an infinite sequence of nodes satisfying four conditions: $s(0)$ says that this infinite sequence of nodes begins with the root R, $(\forall \tau)[s(\tau) \vee s(\tau')]$ that it is indeed a path of the graph; $i(\tau) \equiv s(\tau)$ and $o(\tau) \equiv \overline{s(\tau)}$ define the input and output predicates for the path. This correspondence between formulas and sequence generators is such that a pair of predicates of natural numbers $\langle i, o \rangle$ satisfies the formula if and only if the corresponding two-by-omega matrix of truth values belongs to the behavior of the given sequence generator.

Every formula derived from a sequence generator can be put in the *normal form*

$$(\exists s)\{R[s(0)] \,\&\, ((\forall \tau)P[s(\tau), s(\tau'), i(\tau), o(\tau)])\}, \quad (2)$$

where R and P are both truth functions. Conversely, one can derive from any normal form formula a corresponding sequence generator. There is thus a one-one correspondence between normal form formulas and sequence generators, which correspondence preserves behavior.

The formulas of the sequential calculus can be studied by means of sequence generators, to which combinatorial methods can be easily applied. We will next present a theorem which was arrived at by working with sequence generators but is more easily stated here as a theorem about formulas of the sequential calculus.

We introduce the following definition for a formula $\mathfrak{C}(i, o)$ containing i and o as the only free variables. $\mathfrak{C}(i, o)$ is *uniquely solvable* if and only if for each predicate i there is *exactly one* predicate o such that the pair $\langle i, o \rangle$ satisfies $\mathfrak{C}(i, o)$ in the intended interpretation. It is clear that when a formula $\mathfrak{C}(i, o)$ is uniquely solvable, it *implicitly* defines a func-

tion $\mathfrak{F}$ such that $o = \mathfrak{F}(i)$ if and only if $\mathfrak{C}(i, o)$. $\mathfrak{F}$ is a function which maps the set of all monadic predicates of natural numbers into the set of all monadic predicates of natural numbers.

We are interested in a recursive definition of the type Church (1960) calls an *unrestricted singulary recursion.* Such a recursion is of the form

$$\begin{aligned} s(0) &\equiv A[i(0), \cdots i(h)] \\ s(\tau') &\equiv B[s(\tau); i(\tau'), \cdots i(\tau + h)] \\ o(\tau) &\equiv D[s(\tau)], \end{aligned} \tag{3}$$

where A, B, and D are truth functions and h is a nonnegative integer.

FIRST DEFINABILITY THEOREM: *For every uniquely solvable normal form formula* $\mathfrak{C}(i, o)$, *there exists an equivalent formula* $(\exists s)(\tau)\mathfrak{U}(s, i, o, \tau)$, *where* $\mathfrak{U}(s, i, o, \tau)$ *is an unrestricted singulary recursion. Furthermore, the formula* $(\exists s)(\tau)\mathfrak{U}(s, i, o, \tau)$ *can be constructed effectively from the given formula* $\mathfrak{C}(i, o)$.

The proof of the first definability theorem has three steps.[1] The first step can be expressed if we think of the natural numbers as discrete times with τ being a time variable. Consider now a uniquely solvable normal form formula $\mathfrak{C}(i, o)$ and the corresponding sequence generator with input i and output o. The output o at time τ may, contrary to physical reality, depend on input states which occur at a time later than τ. Let k be the number of nodes of the sequence generator. It turns out that the amount of anticipation is bounded by k^2, that is, the parameter h of the unrestricted singulary recursion is in fact k^2. Thus the first step of the proof of the definability theorem establishes the fact that for a uniquely solvable sequence generator with k nodes, the output o at time τ is independent of input states after $\tau + k^2$.

The second step of the proof of the first definability theorem involves a function or operation on the class of sequence generators, called the subset sequence generator operation, and used by Myhill (1957), Medvedev (1958), and Rabin and Scott (1959). Let Γ be a sequence generator and let Γ^* be *the subset sequence generator of* Γ. The nodes of Γ^* are sets of nodes of Γ. The arrows of Γ^* are placed according to the arrows of Γ in such a way as to take account of the fact that a binary relation induces on the subsets of its domain a function which is single-valued. There are two important facts about Γ^*. First, if Γ is uniquely solvable, then for each predicate i there is exactly one path of Γ^* giving

[1] For the full proof see Sections 3.3 and 4.2 of Burks and Wright (1962).

rise to i. Second, Γ^* had the same behavior as Γ. Previous users of the subset sequence generator operation have shown this second fact to be so when behavior is based on finite sequences. This behavioral equivalence may be extended to our concept of behavior, which is based on infinite sequences, by means of König's (1936) infinity lemma.[2]

As a consequence of steps one and two, our consideration of the normal form formula $\mathfrak{C}(i, o)$ is reduced to a sequence generator Γ^* and a number k^2 having these two properties: first, for each predicate i there is a unique path in Γ^* giving rise to i; second, k^2 is a bound on the time dependence of the predicates s and o on the predicate i, since $s(\tau)$ and hence $o(\tau)$ are independent of the input states after $\tau + k^2$. To obtain the unrestricted singulary recursion for $\mathfrak{C}(i, o)$ we must express $s(\tau')$ as a truth function of $s(\tau)$ and $i(\tau), \cdots, i(\tau' + h)$. For any given i we use the predicate s defined by the corresponding path through Γ^* as the s of the recursion, and we take h to be k^2. A simple argument based on the uniqueness of the path will show that $s(\tau')$ is a time-independent truth function of $s(\tau)$ and $i(\tau'), \cdots, i(\tau' + h)$. To write o as a restricted singulary recursion of i and s is now essentially a matter of an elaborate truth table procedure.

Our definability result may be generalized to arbitrary formulas of the sequential calculus by means of some theorems about this calculus established by J. R. Büchi. Büchi's (1962) main result is that there is a decision procedure for whether or not an arbitrary sentence of the sequential calculus is true in the intended interpretation. An immediate corollary of this result is that there is a decision procedure for unique solvability, since an arbitrary formula $\mathfrak{C}(i, o)$ is uniquely solvable if and only if the following formula of the sequential calculus is true:

$$(i)(\exists o)\{\mathfrak{C}(i, o) \ \& \ (o_1)[\mathfrak{C}(i, o) \supset (\tau)(o(\tau) \equiv o_1(\tau))]\}. \quad (4)$$

The generalization of our definability result involves the concept of a finitely anticipatory formula. A formula $\mathfrak{C}(i, o)$ is *finitely anticipatory* if for each time τ there is an integer h such that the input states from time zero up to time $\tau + h$ determine the output state at time τ. It should be noted that in an unrestricted singulary recursion h is a constant, whereas in the definition of finitely anticipatory, h is a function of τ. Since $\leqq$ can be defined in the sequential calculus, the statement that $\mathfrak{C}(i, o)$ is finitely anticipatory can be expressed by a sentence of

[2] See also Section 2.2 of Burks and Wright (1962).

the sequential calculus. Therefore, the decidability of truth of sentences of the sequential calculus implies that the class of finitely anticipatory formulas is effectively decidable.

SECOND DEFINABILITY THEOREM: *Let* $\mathfrak{C}(i, o)$ *be any formula of the sequential calculus having* i *and* o *as the only free variables. There is an unrestricted singulary recursion* $\mathfrak{U}(s, i, o, \tau)$ *such that* $\mathfrak{C}(i, o) \equiv (\exists s)(\tau)\mathfrak{U}(s, i, o, \tau)$ *if and only if* $\mathfrak{C}(i, o)$ *is both uniquely solvable and finitely anticipatory. Moreover, there is an algorithm applicable to* $\mathfrak{C}(i, o)$ *which yields an equivalent formula,* $(\exists s)(\tau)\mathfrak{U}(s, i, o, \tau)$, *if one exists.*

Roughly speaking, this theorem states that if $\mathfrak{C}(i, o)$ is a formula of the sequential calculus, the relationship between i and o can be expressed by an unrestricted singulary recursion if and only if $\mathfrak{C}(i, o)$ is both uniquely solvable and finitely anticipatory. A complete presentation of the proof of this theorem requires taking Büchi's (1962) proof of decidability for the sequential calculus and also using regular sets as defined by Kleene (1956). For this reason we will not present the proof here, but will make some remarks about it.

The proof uses the *generalized normal form*

$$(\exists s)\{R[s(0)] \,\&\, ((\exists^{\omega}\tau)G[s(\tau)]) \,\&\, ((\forall\tau)P[s(\tau), s(\tau'), i(\tau), o(\tau)]\}, \quad (5)$$

where R, G, and P are truth functions and $(\exists^{\omega})$ is an infinite existential quantifier. "$(\exists^{\omega})G$" means that there exist infinitely many natural numbers satisfying G. The infinite existential quantifier $(\exists^{\omega})$ is definable in the sequential calculus. Büchi has shown that, for any formula $\mathfrak{C}(i, o)$ of the sequential calculus having the predicates i and o as the only free variables, there is an equivalent generalized normal form formula.

The generalized normal form (5) is the same as normal form (2) except for the addition of the conjunct $(\exists^{\omega}\tau)G$. Just as the normal form (2) corresponds to a sequence generator as defined in the beginning of this paper, the generalized normal form (5) corresponds to an extension of the idea of sequence generator in which each node may or may not be labeled as a goal G. For sequence generators with goals, the definition of behavior is modified by requiring in addition that the infinite paths from which the labels are taken, must pass through the set of goal states infinitely often. Figure 4 is an example of a sequence generator with goals.[3] The generalized normal form formula corresponding to

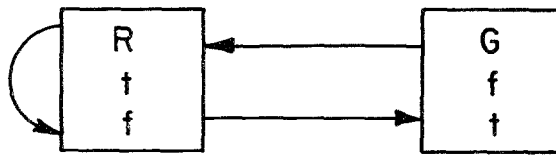

FIG. 4. Sequence generator with goals

Fig. 4 is

$$(\exists s)\{s(o) \ \& \ ((\exists^{\omega}\tau)\overline{s(\tau)}) \ \& \ (\forall\tau)[(s(\tau) \vee s(\tau')) \ \& \ (i(\tau) \equiv s(\tau)) \ \& \ (o(\tau) \equiv \overline{s(\tau)})]\}. \tag{6}$$

The effect of the infinite quantifier $(\exists^{\omega})$ of the formula is obtained in a sequence generator with goals by considering only paths which pass through the goals infinitely many times. There is then a one-one correspondence between generalized normal form formulas and sequence generators with goals, which correspondence preserves behavior.

It was mentioned in connection with the proof of the first definability theorem that in the case of an ordinary uniquely solvable sequence generator with k nodes, the output o at time τ is determined by the input states from time zero up to time $\tau + k^2$. The same is true of those sequence generators with goals which are both uniquely solvable and finitely anticipatory, so that, as before, the amount of anticipation is independent of time.

RECEIVED: April 12, 1962

REFERENCES

BÜCHI, J. R., (1962). On a decision procedure in restricted second-order arithmetic. *In* "Logic, Methodology and Philosophy of Science: Proceedings of the 1960 International Congress," pp. 1–11. Edited by Ernest Nagel, Patrick Suppes, and Alfred Tarski. Stanford University Press, Stanford, California.

BURKS, ARTHUR W., AND WRIGHT, JESSE B. (1962). Sequence generators and digital computers. *In* "Proceedings of Symposia in Pure Mathematics," Vol. 5. Am. Math. Soc., Providence, Rhode Island.

CHURCH, ALONZO, (1960). Application of recursive arithmetic to the problem of circuit synthesis. *In* "Summaries of Talks Presented at the Summer Institute for Symbolic Logic, Cornell University, 1957." Institute for Defense Analysis, Princeton.

[3] See Section 4.4 of Burks and Wright (1962).

KLEENE, S. C., (1956). Representation of events in nerve nets and finite automata. *In* C. E. SHANNON AND J. MCCARTHY (eds.), "Automata Studies," pp. 3–41. Princeton Univ. Press, Princeton, N. J.

KÖNIG, D., (1936). "Theorie der Endlichen und unendlichen Graphen." Akademische Verlagsgesellschaft M.B.H., Leipzig.

MEDVEDEV, I. T., (1958). On a class of events representable in a finite automaton. Translated by J. J. Schorr-Kon from a supplement to the Russian translation of "Automata Studies," C. E. SHANNON AND J. MCCARTHY (eds.). Group Report 34–73, Lincoln Laboratory, Lexington, Mass.

MYHILL, J., (1957). Finite automata and representation of events. *In*: "Fundamental Concepts in the Theory of Systems." WADC Technical Report 57-624, ASTIA Document No. AD 1557 41.

RABIN, M. O., AND SCOTT, D., (1959). Finite automata and their decision problems. *IBM J. Research Develop.* **3**, 114–125.