$\Pi(f)$ and a suitable code for the number n. (Both f and n are variables here.) Then the machine is to go to work and is to produce a code for a number if and only if $f(n)$ is defined; and if $f(n)$ is defined the coded number is to be $f(n)$. (2) Similar to (1) but now the machine is to reproduce on the tape not only $f(n)$ but (in coded form) all steps that the machine with the program $\Pi(f)$ would go through in computing $f(n)$ from n.

The author asserts that problem (1) can be solved because, using the standard notation and terminology for recursive functions introduced by Kleene, f has a Gödel number z and if $f(n)$ is defined $f(n) = U(\mu_y T_1(z, n, y))$. So if one constructs the program for B-computing $U(\mu_y T_1(z, n, y))$ one has solved (1). Problem (2) is said to be more difficult and the second paper is mostly devoted to constructing in detail a program solving (2). It seems to the reviewer, however, that since the computation that machine B goes through when programmed by $\Pi(f)$ and fed n on the tape is also a partial recursive function of $\Pi(f)$ and n, (2) could be settled by a short argument like that used in (1). STEVEN OREY

ALONZO CHURCH. *Application of recursive arithmetic to the problem of circuit synthesis* ***Summaries of talks presented at the Summer Institute for Symbolic Logic Cornell University, 1957,*** 2nd edn., Communications Research Division, Institute for Defense Analyses, Princeton, N. J., 1960, pp. 3–50, 3a–45a.

In this early paper on the application of formal logic to circuits and automata Church develops and extends a theory of restricted recursive arithmetic first presented in his review (XX 286) of an article by E. C. Berkeley. The author here presents several alternatives for the recursion schemata of that system: Let $s = (s_1, \ldots, s_k)$, $r = (r_1, \ldots, r_n)$. The schemata for restricted recursion (A) are

$$r_i(0) = P_i[s(0)] \qquad i = 1, \ldots, n$$
$$r_i(t + 1) = Q_i[s(t), s(t + 1), r(t)]$$

and for wider restricted recursion (B) are

$$r_i(0) = P_{i0}[s(0)], \ldots, r_i(h) = P_{ih}[s(0), \ldots, s(h)]^{\cdot}$$
$$r_i(t + h + 1) = Q_i[s(t), \ldots, s(t + b), r(t), \ldots, r(t + h)];$$

for unrestricted singulary recursion (C), the schemata are the same as for (B) except that $r_i(j) = P_{ij}[s(0), \ldots, s(a)]$ for $j = 0, \ldots, h$.

It is shown that circuits and finite automata can be treated by means of restricted recursive arithmetic (A), or by (B), which is reducible to (A).

Two problems for recursive arithmetic are studied. The *synthesis problem*: given a requirement $S(t)$ in a logical system which is an extension of recursive arithmetic, to find (if possible) recursion equivalences for a circuit which satisfies the requirement. And the *decision problem*: given both requirement and recursion equivalences, to determine whether the circuit satisfies the requirement.

The synthesis problem is solved first for the case of one free variable (Case 1):

$$S[i_j(0), \ldots, i_j(a_j), i_j(t), \ldots, i_j(t + a_{2j}), o_l(0), \ldots, o_l(b_l), o_l(t), \ldots, o_l(t + b_{2l})]$$
$$j = 1, \ldots, \mu, \quad l = 1, \ldots, \gamma$$

The synthesis method contains within it a test to determine if a solution is possible.

An alternative treatment of this synthesis problem is included by Wang in XXV 373.

Case 2 of the synthesis problem, a requirement with two or more free variables is solved by reduction to Case 1. Case 3, a requirement with quantifiers, is treated by using Behmann's procedure to limit the scope of each quantifier. Several subcases of Case 3 are solved, but the complete solution is not obtained. Case 4, in which the requirement includes $=$ and $<$, is proposed and a conjecture given about a possible method of solution. An additional case, in which the requirement contains $+$, $=$ and

quantifiers, has been solved negatively by Büchi, Elgot, and Wright (*The nonexistence of certain algorithms of finite automata theory,* ***Notices of the American Mathematical Society,*** vol. 5 (1958), p. 98, Abstract).

The decision problem, to determine if circuit $E(t)$ satisfies a requirement S, was solved for Case 1 by the reviewer (XXII 337) and McNaughton has observed (XXIV 241) that the solution extends to Case 2. Church provides here an alternative solution, by reducing the problem to the question of whether a circuit satisfying both $E(t)$ and S can be synthesized. In addition he solves the more difficult case in which S is of the form $(t)H(t) \supset (t)A(t)$. This makes it possible to solve also the decision problem for Case 3. The decision problem for Case 4 follows from a recent result of Büchi (XXVIII 100).

The author provides valuable examples showing the application of the formal system to circuits and automata. He is careful to point out that the simplification problem remains open even where the synthesis problem is solved.

By generalizing to n-ary predicates, a system of propositional recursive arithmetic is obtained, which, it is argued, represents potentially infinite automata. Open questions in this area are presented.

A complete historical note is appended, relating the paper to earlier works, in particular to the work of McCulloch and Pitts, Burks and Wright, and Kleene. More recently, an account of developments on these problems has been given by Church in *Logic, arithmetic, and automata,* ***Proceedings of the International Congress of Mathematicians 1962,*** pp. 23–35.

Author's correction: On page 11a, after the word "for" in line 11, insert "t', t", t"', . . . , and 1, 2, 3, . . . are used as substitute notations for." JOYCE FRIEDMAN

SHELDON B. AKERS, Jr. *A truth table method for the synthesis of combinational logic.* ***IRE transactions on electronic computers,*** vol. EC-10 (1961), pp. 604–615.

A truth-function which can be expressed by a normal formula containing no negation signs is called a *frontal function*; and a truth-function which can be expressed by a normal formula in which no letter occurs both affirmed and negated is called a *unate function.* The special properties of frontal and unate functions have been discussed extensively in the literature (cf. XIX 142 and Robert McNaughton, *Unate truth functions,* ***IRE transactions on electronic computers,*** vol. EC-10 (1961), pp. 1–6). In the paper being reviewed, Akers presents a technique for modifying the truth-table of an arbitrary truth-function so that it has the properties of the truth-table of a unate function. This is done by adding a column for the negative of each letter of the function and then treating these columns as corresponding to independent truth-variables. Rules are given for rewriting the truth-table in a more compact form by eliminating some of the rows and columns. This is possible because of the special properties that a truth-table for a unate function must have. Finally procedures for deriving economical circuits from the simplified truth-table are discussed.

The techniques presented are most useful for truth-functions for which the truth-value is not specified for many of the rows of the truth-table (incompletely specified functions) and as such must be considered in relation to the other techniques for treating such functions (E. J. McCluskey, Jr., *Minimal sums for Boolean functions having many unspecified fundamental products,* ***Transactions of the American Institute of Electrical Engineers,*** vol. 81 part I (1962), pp. 387–392). Since the author does not give any detailed algorithm for implementing his rules or any estimates of the efficiency of his techniques it is not possible to make such a comparison on the basis of the information available. The ideas presented in this paper are interesting: the real value of the work will become clear only after a computer program has been written to test these ideas. E. J. MCCLUSKEY