Mathematics

1958

SovietRxiv

Abstract

Full Text

Mathematics

B. A. Trakhtenbrot

Synthesis of Logical Networks Whose Operators Are Described by Means of the Calculus of Unary Predicates

(Presented by Academician M. V. Keldysh on 18 July 1957)

1°. In the synthesis of truth-functional logical networks (automata without memory), the desired mode of operation (the network operator) can always be described by means of a finite functional table (matrix) indicating the dependence of the output information on the input information. If the input and output signals are represented in binary code, then from the given matrix one directly reconstructs a system of functions of the algebra of logic which, in accordance with the basic idea of Shestakov–Shannon, is then transformed into the logical network that realizes it.

When considering a logical network with memory, the natural analogue of a finite matrix is already an infinite tree representing the dependence of the output information on the input information in a process continued without bound in time. Since the direct specification of an infinite tree is already impossible, one usually seeks to reduce the matter to consideration of finite parts of the tree, in particular by the purposeful selection of inclusion tables (⁀2). In this connection there already arises such a specific problem as the preliminary estimation of the number and assignment of intermediate delay elements necessary for the proper organization of the network and its memory. The main difficulty here consists in the fact that, in the initial verbal description of the desired mode of operation, these elements, as a rule, do not appear, and each time they must somehow be discovered.

In connection with this, it is of interest to find means that would make it possible to carry out the synthesis of a logical network with memory directly from the initial description of the operator, given in terms of some sufficiently precise formal language. In the present article this problem is considered as applied to an extended calculus of unary predicates.

2°. We shall use the slightly modified notation and terminology of (⁀1), adapted to such logical networks in which information is supplied through n input binary channels and is issued through one binary channel. By an $\mathcal{L}$-operator (an operator realizable in a logical network) we shall understand an operator

$$Z(t) = T[X_1, \dots, X_n; t], \tag{1}$$

which transforms a system of input predicates $\{X_i(\tau)\}$ (time $\tau = 1, 2, \dots$) into an output predicate $Z(\tau)$, and which can be specified by means of the canonical scheme

$$\begin{gathered} Z(t) = \Phi[X_1(t), \dots, X_n(t), \Gamma_1(t), \dots, \Gamma_k(t)], \\ \Gamma_\nu(t+1) = \Psi_\nu[X_1(t), \dots, X_n(t), \Gamma_1(t), \dots, \Gamma_k(t)], \\ \Gamma_\nu(1) = \sigma_\nu \quad \text{(initial state)}, \end{gathered} \tag{2}$$

where Φ, Ψ_ν are functions of the propositional calculus; σ_ν are constants (0 or 1); $\nu = 1, \dots, k$.

Let us note that instead of the scheme (2) one could specify the operator (1) by means of the following formula (3), in which $\Gamma_\nu^{\sigma_\nu}$ denotes Γ_ν when $\sigma_\nu = 1$ and $\overline{\Gamma}_\nu$ when $\sigma_\nu = 0$, while $\sim$ is the equivalence sign:

$$(E\Gamma_1) \dots (E\Gamma_k)\{\Phi[X_1(t), \dots, \Gamma_k(t)] \mathop{\&}_{\nu=1}^{k} \mathop{\&}_{\tau<t} [\tau = 1 \& \Gamma_\nu^{\sigma_\nu}(\tau) \vee \mathop{\vee}_{\sigma<\tau} (E\sigma)(\sigma = \tau - 1 \& (\Gamma_\nu(\tau) \sim \Psi_\nu[X_1(\sigma), \dots, \Gamma_k(\sigma)])) \tag{3}$$

The form of this formula, in which, besides the operations of the propositional calculus, quantifiers also occur, leads to the description of the following formal language.

Formulas. The formulas of the language are declared to be ordinary formulas of the extended calculus of unary predicates (with predicate and individual quantifiers), with the sole special feature that the individual quantifiers are restricted, i.e. have the form $[x]$, where $[x]$ denotes (x) or (Ex);

$$x < y$$

x, y are arbitrary individual variables (substantively ranging over the values $1, 2, \dots$ of discrete time), and $<$ denotes $<$ or $\leqslant$.

t-**formulas** are formulas possessing two properties:

1) in them there occurs exactly one free variable t;

2) all individual quantifiers are t-controlled.

The notion of t-controlledness is defined inductively:

I. Quantifiers $[x]$ are t-controlled.

$$x < t$$

II. If $[\tau]$ is a t-controlled quantifier, then all quantifiers of the form

$$\tau < \sigma$$

$[x]$, situated in its scope, are also t-controlled.

$$x < \tau$$

III. There are no other t-controlled quantifiers.

We introduce the notation $\mathfrak{A}(X_1, \dots, X_n; t)$ for a t-formula containing exactly the free predicate variables $X_1, \dots, X_n$ (where this causes no misunderstanding, we shall also write $\mathfrak{A}$ or $\mathfrak{A}(t)$). Under any fixed interpretation of these predicates with a natural argument, the formula also determines a completely definite predicate of a natural argument, and in this sense it describes an operator transforming the system of predicates $\{X_i\}$ into one predicate.

Theorem 1. Every $\mathcal{L}$-operator is described by a t-formula.

This follows directly from the fact that the equalities $\tau = 1$ and $\sigma = \tau - 1$, occurring in (3), can be expressed in our formal language.

3°. The converse assertion also holds:

Theorem 2. Every t-formula $\mathfrak{A}(X_1, \dots, X_n; t)$ describes an $\mathcal{L}$-operator acting on the system of n predicates $\{X_i\}$.

The proof of this theorem consists in the description of an algorithm which is applicable to any t-formula and transforms it into the canonical scheme of the corresponding operator. Since the passage from a canonical scheme to the logical network realizing it is carried out effectively by a known method (see, for example, ([1])), the algorithm just mentioned is essentially an algorithm for synthesizing the $\mathcal{L}$-operator described by the t-formula. We first introduce some notation and concepts.

Regular t-formulas. A t-formula is called **regular** if in it: I —there are no occurrences of the free variable t as arguments of predicates; II —there are no quantifiers of the form $[x]$. This means that in the formula there are occurrences of the symbol t only in quantifiers of the type $[x]$.

$$x \leqslant t$$

$$x < t$$

Companions of a t-formula. A t-formula $\mathfrak{B}$ is called a **companion** of a t-formula $\mathfrak{A}$ if every object or predicate symbol occurring in $\mathfrak{B}$ also occurs in $\mathfrak{A}$, and its occurrences in $\mathfrak{A}$ and in $\mathfrak{B}$ are simultaneously free or bound. Let $\mathfrak{B}(t)$ be some regular t-formula. By $\mathfrak{B}(t+1)$ we shall denote the t-formula obtained from $\mathfrak{B}(t)$ by replacing every quantifier of the form $\underset{x<t}{[x]}$ by the quantifier $\underset{x\leq t}{[x]}$. Obviously, $\mathfrak{B}(t+1)$ is a companion of $\mathfrak{B}(t)$, but $\mathfrak{B}(t+1)$ is no longer regular (condition II is violated).

The synthesis algorithm consists of operations of two types:

A. **Expansion in regular companions.** Every t-formula $\mathfrak{A}[X_1, \dots, X_n; t]$ is transformed into a companion of the form

$$\Phi[X_1(t), \dots, X_n(t), \mathfrak{B}_1(t), \dots, \mathfrak{B}_k(t)],$$

where Φ is a function of the propositional calculus, and the $\mathfrak{B}_i$ are certain regular companions of the formula $\mathfrak{A}$.

B. **Computation of the value of a regular formula** $\mathfrak{B}(t)$ for $t = 1$; this value is a constant (0 or 1), depending on the free predicates of the predicate variables of the formula $\mathfrak{B}$.

Operation A, in turn, is a chain of identical transformations of the extended calculus of monadic predicates, as well as **lowerings of quantifiers**. By lowering a quantifier is meant replacing an expression of the form $\underset{x\leq t}{(x)}\,\mathfrak{A}$ by the expression $\underset{x<t}{(x)}\,\mathfrak{A}\ \&\ \mathfrak{A}_x^t$, where $\mathfrak{A}_x^t$ denotes the result of substituting in $\mathfrak{A}$ the variable t for the variable x (similarly for existential quantifiers).

The algorithm works as follows. As a result of the successive application of operation A to the initial formula $\mathfrak{A}$, and also to formulas of the form $\mathfrak{B}_i(t+1)$, where the $\mathfrak{B}_i(t)$ are certain regular companions of the formula $\mathfrak{A}$ arising in the course of the process, a finite system of expansions in regular companions is generated:

$$\begin{aligned} \mathfrak{A}(t) &= \Phi[X_1(t), \dots, X_n(t), \mathfrak{B}_1(t), \dots, \mathfrak{B}_k(t)], \\ \mathfrak{B}_i(t+1) &= \Psi_i[X_1(t), \dots, X_n(t), \mathfrak{B}_1(t), \dots, \mathfrak{B}_k(t)], \end{aligned} \tag{4}$$

$i = 1, \dots, k$. Replacing in (4) the expressions $\mathfrak{A}(t)$, $\mathfrak{B}_i(t)$ respectively by the predicate symbols $Z(t)$, $\Gamma_i(t)$ and computing the initial values $\Gamma_i(1)$ (operation B), we obtain the canonical circuit of the $\mathcal{L}$-operator.

4°. The algorithm described is in fact applicable also to a language broader than the one indicated above. A detailed consideration of this question, as well as the derivation of quantitative estimates characterizing the economy of the proposed algorithm and of the canonical circuit it generates, is not part of the aim of the present article. We confine ourselves to one remark: if the language is

extended by admitting "two-sided" bounded quantifiers of the form $\underset{y<x<z}{[x]}$, with the natural generalization of the notion of t-controllability, then the algorithm will work as before.

Let us illustrate this by the following example: let

$$\mathfrak{A} \underset{\text{Df}}{=} (\sigma)\{X_1(\sigma) \vee (E\tau)[X_2(\tau)\ \&\ (\rho)\, X_3(\rho)]\}. \tag{5}$$

$$\sigma<t \qquad \sigma<\tau<t \qquad \sigma<\rho<\tau$$

The initial formula is already regular; therefore we apply operation A to the formula $\mathfrak{A}(t+1)$. Lowering the quantifiers gives

$$X_1(t)\ \&\ (\sigma)\{X_1(\sigma) \vee (E\tau)[X_2(\tau)\ \&\ (\rho)\, X_3(\rho)] \vee X_2(t)\ \&\ (\rho)\, X_3(\rho)\}. \tag{6}$$

$$\sigma<t \qquad \sigma<\tau<t \qquad \sigma<\rho<\tau \qquad \sigma<\rho<t$$

Further identical transformations of the predicate calculus

transform (6) into the regular expansion for $\mathfrak{A}(t+1)$

$$X_1(t)\&[\mathfrak{A}(t) \vee X_2(t)]\&\mathfrak{B}(t), \tag{7}$$

where $\mathfrak{B}(t)$ is an abbreviated notation for the regular companion

$$\underset{\sigma<t}{(\sigma)}\left\{X_1(\sigma) \vee \underset{\sigma<\tau<t}{(\mathrm{E}\tau)}\left[X_1(\tau)\&\underset{\sigma<\rho<\tau}{(\rho)}\, X_3(\rho)\right] \vee \underset{\sigma<\rho<t}{(\rho)}\, X_3(\rho)\right\}.$$

In an analogous way we obtain the expansion for the formula $\mathfrak{B}(t+1)$:

$$[\mathfrak{A}(t) \vee X_2(t) \vee X_3(t)]\&\mathfrak{B}(t). \tag{8}$$

After substituting the initial values (operation B), we obtain the canonical circuit

$$\begin{aligned} Z(t+1) &= X_1(t)\&[Z(t) \vee X_2(t)]\mathfrak{B}\Gamma(t), \\ \Gamma(t+1) &= [Z(t) \vee X_2(t) \vee X_3(t)]\&\Gamma(t), \\ Z(1) &= 1, \qquad \Gamma(1) = 1. \end{aligned} \tag{9}$$

5°. Further extension of the formal language in order to facilitate the formulation of the initial descriptions of operators is of interest; however, it must be taken into account that, in general, the synthesis algorithm then becomes more complicated, and even the phenomenon of algorithmic undecidability may arise.

Indeed, let Ω be some formal language; for its formulas the following two algorithmic problems naturally arise:

1. To determine whether a formula $\mathfrak{A}$ specifies an $\mathcal{L}$-operator or not (a recognition algorithm).
2. For a formula $\mathfrak{A}$ specifying an $\mathcal{L}$-operator, to construct the corresponding canonical circuit (a synthesis algorithm).

If the language Ω admits ordinary schemata of recursive definitions (even only primitively recursive ones), then both problems are algorithmically undecidable.

6°. In conclusion, let us note that the results of the present paper can be interpreted in terms of the representability of events in finite automata ([3],[4]). In essence, the operation $S_i \to S$, introduced by Yu. T. Medvedev ([4]), is equivalent to the existential quantifier over a predicate variable, but the formal language that can be constructed from the operations of Yu. T. Medvedev is narrower than the language described in the present paper.

Penza State Pedagogical Institute
named after V. G. Belinsky

Received
16 VII 1957

References Cited

[1] B. A. Trakhtenbrot, *DAN*, 112, No. 6 (1957).
[2] M. A. Gavrilov, *Theory of relay-contact circuits*, Publishing House of the USSR Academy of Sciences, 1950.
[3] S. C. Kleene, in: *Automata*, IL, 1956.
[4] Yu. T. Medvedev, ibid.

Note: Figure translations are in progress. See original paper for figures.