The synthesis of logical nets consisting of NOR units

By H. P. Williams*

This paper describes an algorithm for synthesising a logical net consisting of NOR units. Starting with a logical function presented as a truth table the function is converted into a succession of NOR statements. A simplifying procedure is used which, while not always resulting in the minimum number of NOR units, produces an economical solution. Details are given of how this algorithm can be programmed for automatic computation.

(First received November 1967)

In the construction of electronic and fluidic circuits it is often necessary to construct a logical net to perform some given logical function. These nets are often synthesised from NOR units. The NOR unit acting on two inputs, A and B, performs the function $\overline{A \vee B}$ written in Boolean algebra. This is a 'universal function' in the sense that any function in Boolean algebra can be constructed by using successive applications of this function only. Hence the advantage of using NOR units in a logical net is that no other type of logical unit is needed.

A logical net to perform some prescribed logical function can usually be constructed in many different ways. Clearly it will usually be desirable to construct such a net using as small a number of components as possible, in this case NOR units. An algorithm is described which, starting with a logical function presented as a truth table, converts the function to an expression composed only of NOR statements. Simplifications are performed which result in the use of an economical number of NOR units. This algorithm has been programmed for automatic computation.

Description of the algorithm

The method is based on successive applications of operations described by Quine (1955). First the function under consideration is presented as a truth table. The truth table is always written in the way shown below using 0 to signify 'false' and 1 to signify 'true'. The rows represent successive numbers written in binary form. **Table 1** gives an example of a function F of three arguments A, B, C.

Table 1

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Stage 1. Delete the rows of the truth table which correspond to the function having value 1. In the example these are rows 1, 7 and 8.

Stage 2. Simplify the remaining portion of the truth table using the following three operations based on those of Quine.

(i) If any two rows differ only in one column, in one row the entry being 0 and in the other 1, delete one of the rows and delete the entry in the other row. In the example after performing this operation on rows 2 and 4 the result would be the single row 0 – 1.

This operation is based on the logical equivalence,

$$\bar{X} \cdot \phi \vee X \cdot \phi \equiv \phi$$

(ii) If one row completely contains another row except for a difference in one column, where one entry is 0 and the other entry is 1, delete this entry in the longer row. For example in the following two rows the first entry in the second row would be deleted.

0 – 1
1 0 1

This operation is based on the logical equivalence

$$X \vee \bar{X}\phi = X \vee \phi$$

(iii) If one row completely contains another delete the longer row. For example in the following two rows the second row would be deleted.

0 – 1
0 0 1

This operation is based on the logical equivalence

$$X \vee X \cdot \phi \equiv X$$

Each pair of rows is examined in turn and operations (i), (ii) or (iii) performed if possible. If one of the operations is performed the comparison of all the rows is repeated. After completion of this stage the function can be represented as a negation of a disjunction of the complete sum of prime implicants. In the example in Table 1 the result is

$$\overline{\bar{A} \cdot C \vee \bar{A} \cdot B \vee A \cdot \bar{B} \vee \bar{B} \cdot C}$$

* *Department of Mathematics, The University, Leicester.*

Stage 3. Examine the resulting rows of the simplified truth table for a row containing only a 1.

Case (*i*). If such a row exists the input represented by the column of this single entry is an input to the last NOR unit in the net. The logical function can now be considered in the form

$$\overline{X \vee \phi}$$

where X is the input which has just been considered and ϕ is the rest of the function under the negation. The function can also be written in the form

$$\text{NOR}\ (X, \phi)$$

showing more clearly that X is an input to the last NOR unit. The other input to this NOR unit must be a net performing the logical function ϕ. After deleting the row of the simplified truth table containing the single entry 1, if no rows of the truth table remain then this NOR unit has no other inputs. If only one row remains and this consists of only an entry 1, the column of this entry gives the other input to the NOR unit. Otherwise the remainder of the truth table is expanded into the standard form. This can be done by comparing the entries in each row of the remainder of the truth table with the corresponding entries in each possible row of the standard truth table. Where these corresponding entries are equal the row of the standard truth table is retained. For example if the remainder of the truth table consisted of the single row $-\ 0\ -$, on comparison with the standard truth table shown in Table 1 it can be seen that the second column in rows 1, 2, 5 and 6 is 0. Hence these rows are retained and the row $-\ 0\ -$ has been expanded into the four rows:

0 0 0
0 0 1
1 0 0
1 0 1

These rows are now deleted from the standard truth table such as Table 1. The remaining truth table is now simplified as in stage 2 and the whole procedure repeated producing a logical net for the function ϕ which connects onto the last NOR unit.

Case (*ii*). If no row containing only a 1 exists then none of the external inputs goes to the last NOR unit. This is the case in the example where the function has been expressed in the form

$$\overline{\bar{A}\,.\,C \vee \bar{A}\,.\,B \vee A\,.\,\bar{B} \vee \bar{B}\,.\,C}$$

In this case the rows of the simplified truth table are split up into two sections if possible so the function can be considered in the form

$$\overline{\phi \vee \psi}$$

or alternatively in the form

$$\text{NOR}\ (\phi, \psi).$$

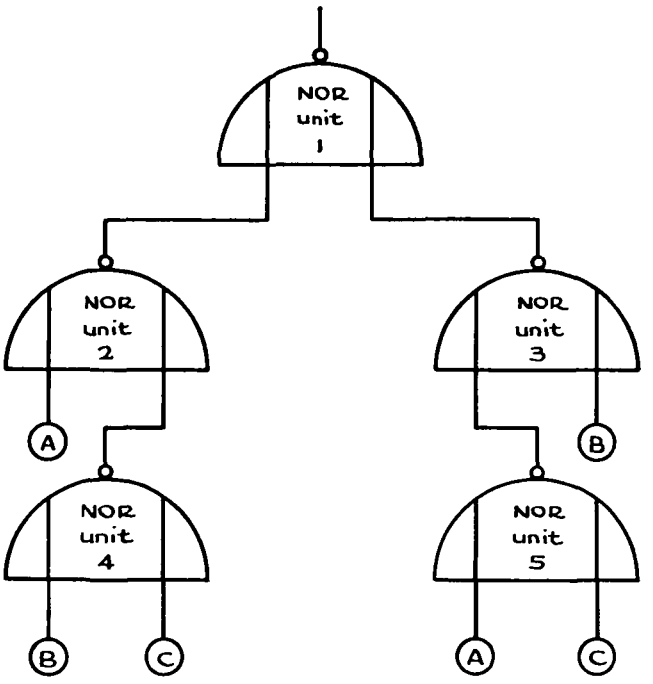

Fig. 1

The inputs to the last NOR unit must therefore be logical nets representing the functions ϕ and ψ. These functions are considered separately. Each section is therefore considered one at a time. The rows of the first section are expanded as in case (*i*) above. These rows are then deleted from the standard truth table which is then simplified as in stage 2, and the whole procedure is repeated producing a logical net for the function ϕ which is connected to the last NOR unit. The function ψ is considered in a similar manner. If it is not possible to split the rows of the simplified truth table into two sections, i.e. we have only one row left, one of the inputs to the last NOR unit is left blank and this single row considered as the other section and treated as before.

After repeating these procedures a sufficient number of times the whole function is represented as a net of NOR units.

The example given above is now considered in detail. After stage 1 and stage 2 have been performed for the first time the resulting truth table is

0 − 1
0 1 −
1 0 −
− 0 1

This shows that the function can be represented in the form

$$\overline{\bar{A}\,.\,C \vee \bar{A}\,.\,B \vee A\,.\,\bar{B} \vee \bar{B}\,.\,C}$$

Since no rows have only a 1 as entry NOR unit 1 as shown in **Fig. 1** has no external inputs. The inputs are

logical nets representing the functions ϕ and ψ where

$$\phi = \bar{A} \,.\, C \vee \bar{A} \,.\, B$$

$$\psi = A \,.\, \bar{B} \vee \bar{B} \,.\, C$$

The function ϕ is represented by the first section of the truth table which is

0 – 1
0 1 –

These rows are expanded to give the following rows

0 0 1
0 1 1
0 1 0

When these rows are deleted from the standard truth table and the resulting truth table simplified the result is

– 0 0
1 – –

This shows that NOR unit 2 has A as an input. The first row of this truth table is expanded and the rows deleted from the standard truth table. After simplification the result is

– – 1
– 1 –

This shows that NOR unit 4 has two external inputs, B and C, and that no more NOR units connect into NOR unit 4.

The function ψ is now considered. This is represented by the second section of the first truth table which is

1 0 –
– 0 1

When these rows are expanded and deleted from the standard truth table, after simplification the following truth table results:

0 – 0
– 1 –

This shows that NOR unit 3 has B as an input. The first row of the truth table is expanded and deleted from the standard truth table. After simplification the following truth table results:

– – 1
1 – –

This shows that NOR unit 5 has two external inputs A and C, and that no more NOR units connect into NOR unit 5. The net is therefore completed.

It is clearly not often practical to synthesise logical nets by performing these procedures manually. Using a computer, however, the synthesis can be performed very rapidly. The initial data for such a computation need only be a number specifying the number of inputs being considered (in the example this number is 3) and the numbers of the rows of the standard truth table corresponding to the function being considered having value 1. This algorithm has been programmed and details of this are now given.

Programming the algorithm for computation

In order to compute a net it was found convenient to consider a maximum net in which two NOR units connect to each NOR unit in the net. This net is made sufficiently large that any possible net would be a proper part of it. Each NOR unit in the maximum net is numbered in a standard way. This serves as a useful framework. A cycle of the algorithm is performed for each NOR unit in the final net. After the completion of each cycle the computation moves on to consider a NOR unit with a higher number, or if this particular branch of the network has been completed it goes back to a lower number on the branch and then ascends to a higher number on another branch.

It is necessary to store certain numerical arrays. The main arrays are now described.

(i) An array consisting of 0s and 1s such as Table 1 where each row represents a successive number in binary form. For a computation of a net with n external inputs this array would have dimensions $n \times 2^n$. At each cycle in the algorithm certain rows of this table are 'deleted' by overwriting with other figures and the remainder of the table simplified using the three operations described.

(ii) Each NOR unit in the net can be regarded as the last NOR unit in some subnet performing a certain logical function. Therefore, associated with each NOR unit there must be a representation for this logical function. This is done by means of an array listing the numbers of the rows of the standard truth table which correspond to the function having value 1. Since the rows of the standard truth table are successive binary numbers the numbers of the rows can easily be computed. As there is a one-dimensional array associated with the number of each NOR unit the total array is two-dimensional.

(iii) There can be up to two external inputs to each NOR unit. These external inputs are numbered. Associating these two numbers with the number of each NOR unit gives a two-dimensional array.

(iv) One of the dimensions of the array (ii) will vary with the number of each NOR unit considered. Associating this dimension with the number of each NOR unit gives a one-dimensional array.

The program was written in FORTRAN IV and run on an IBM 360 computer with a core storage of 64K. It was found convenient to limit the program to synthesising nets with up to 8 external inputs, i.e. dealing with logical functions of up to 8 variables. Large amounts of core storage would have been used if all the arrays had been stored in core. Array (ii) was therefore stored by writing each row of it as a record on a magnetic disc. Since it was only necessary to read a record for each NOR unit and to write up to two records for each

NOR unit the extra time taken was small. To deal with functions of many more than 8 variables would probably have necessitated also writing array (i) on disc.

After compilation the amount of time taken for execution of the program was quite short, not being more than ten minutes for a net with 8 external inputs, and very much shorter for nets with a lesser number of external inputs.

Acknowledgement

The author would like to thank Mr. Brian Foster of IBM for help in writing the program.

Reference

QUINE, W. V. (1955). 'A way to simplify truth functions', *American Math. Monthly*, Vol. 62, No. 9, pp. 627–31.

Book Review

Sequential Machines and Automata Theory, by TAYLOR L. BOOTH, 1967; 592 pages. (New York, London, Sydney: *John Wiley and Sons Inc.*, 162s.)

According to the preface of this book it was written 'to provide a unified treatment of sequential machines and automata theory and their interrelationships' and uses 'an engineering rather than a formal mathematical style of presentation'.

What is 'engineering style'? Presumably it is a style of writing in which one is not presented with highly formal definitions and detailed rigorous proofs but a style which concentrates on basic ideas and motivations, which gives ideas of proofs which will stand up if one considers them more deeply and which does not formalise for the sake of formalisation. A superb book written in such a style and covering much (but not all) of the material in Booth's book is 'Computation, Finite and Infinite Machines', by M. Minsky (see review in this *Journal*, Vol. 10, p. 391). Unfortunately many of the informal statements of Booth, whilst it is easy for someone knowledgeable in the subject to see what he intends to say, are rather vague and all too easily collapse if examined closely. There should be more explanation of the concepts involved and less of a rush into formalisations in the 'engineering approach' of this book.

This is unfortunate as the book could have been very good indeed. It has wide coverage (mathematical preliminaries; finite state machines, their decomposition, minimalisation and identification; regular expressions; Turing machines; recursive functions and computability problems; phrase structure grammars; Markov processes and probabalistic machines). It is well planned with informal introductions, more formal definitions, relations to other systems and key properties, summaries, good bibliographies and examples. It has illustrative diagrams, charts and tables. It is a well produced book. Printing errors exist but are rare.

Some specific criticisms follow. The definitions of, and relations between, computable, partially computable, algorithm and program (p. 360) are vague. No mention is made of why such an apparently restrictive definition of 'computable' is used, i.e. of Church's Thesis. At the end of p. 361 the following sections are motivated as 'defining the relationship between Turing machines and computable functions'. As a computable function has been defined as one which a Turing machine can compute this is a strange motivation. On p. 407 it is stated that there is a strong connection between finite state machines and finite state languages because sentences of such languages are generated from left to right; this is vague, which would not matter so much except that attempts to make it precise could lead to incorrect conclusions. Generally the 'proofs' that a certain machine cannot perform a given task only consider one possible way the machine might work; a remark should be added that it can be proved that any way one thinks of will not work. Regular expressions are motivated as 'describing the behaviour of' sequential machines. How do they describe behaviour? Moreover sequential machines are defined as possibly having an infinite set of states whereas regular expressions are related to finite state automata. In the definition of semigroup on p. 32 the fact $c \epsilon S$ should not be a hypothesis and the equations $a \circ (b \circ c) = (a \circ b) \circ c = a \circ b \circ c$ are confusing without a comment that the last expression is merely an abbreviation for either of the first two. An example on p. 33 is described as 'illustrating the usefulness' of a certain procedure, but does no such thing. It is untrue that A. M. Turing was 'not interested in the design of information-processing devices' (p. 353).

These are all quibbles, but many such remarks could be made about statements in the book. The book is a useful one for a lecturer to possess, and to guide students in their reading of it, but not a book to be read without such guidance or without also referring to other books on the subject which give a better feel for the subject.

D. C. COOPER (Swansea)