REVERSAL IN BOOLEAN MINIMALIZATION

By Mitchell O. Locks

ABSTRACT

A computerizable Quine-type algorithm is presented for minimalizing Boolean polynomials. It differs from other minimalization methods by the fact that it incorporates a «reversal» technique. Forward (minimalizing or polynomial reducing) sequences are alternated with reversal (partial expansion) sequences in a search or a unique minimal form. One of three decisions results from each reversal sequence and subsequent reduction: the minimal polynomial is unique; there are alternative minimal forms; or further reduction is possible.

The algorithm includes arrangement of terms, and the sequencing of steps and pairwise matching of terms in both the forward mode and the reversal mode.

INTRODUCTION

Boolean polynomials are widely used in fields such as logic, electrical engineering, fluid power mechanics and data structures to characterize logical circuits and sequences. Their uses in statistics are limited to areas such as theoretical probability, pattern recognition and system reliability. These applications, however, appear to be growing.

A Boolean polynomial is for a binary-valued system, f, which is a function of n binary-valued components x_i, $i = 1, \ldots, n$, not all necessarily independent of one another. The binary values represent the presence ($f_i = 1$, or $x_i = 1$) or absence $f_i = 0$, or $x_i = 0$) of a signal or characteristic which defines the system, or else the component. Three-valued logic is also covered, in the sense that if f or x_i is either zero or one, it is specified; if it is not specified, then it is «free» to assume either value. Thus, each component has three values, zero, one, or unspecified (frequently called «don't care»).

The universal set u is the set of 2^n binary n-vectors, each vector denoting a different possible state of the system. A Boolean polynomial represents a subset of u, called a *lattice*. Each term (also called «minterm») is for a complete (i.e., including both the g.l.b. and the l.u.b.) subset, also called *sublattice*, with m fixed-valued elements, $m \leqq n$, common to all of the 2^{n-m} elements (binary n-vectors) in that subset. In this discussion, «term,» «complete subset,» and «sublattice» are used interchangeably where the meaning is clear from the context, as well as «variable,» «component,» and «indicator.»

It may appear at times that «polynomial» and «lattice» are interchangeable, but this is not the case. A lattice is a set, and a polynomial a way of representing the set. There can be many polynomials for any given lattice. Indeed, the objective of minimalization is to find the «best» or shortest polynomial out of all alternatives available.

Minimalization describes a lattice with the smallest possible number of terms, and the least average number of indicators per term. Stated another way, this is the same as finding the largest sublattices, none of them proper subsets of one another, which completely cover the lattice. In digital logic circuits, for example, minimalization results in the smallest hardware requirement, in data retrieval the smallest search.

Historically, all of the significant work in minimalization of Boolean polynomials emanates from a 1952 paper by Quine [1]. Quine's technique has been modified by McCluskey [2], who employed a decimal coding scheme on the binary-number equivalents of the terms. A number of texts in logical design, examples [3, 4, 5] describe the so-called Quine-McCluskey technique, which is essentially the McCluskey modification.

In this paper, we do not employ the McCluskey modification, but use instead a straightforward extension of the Quine method without any intervening decimal number equivalents. However, rather than requiring a full extension of the polynomial to its «developed form,» minimalization can be performed on any polynomial without any expansion of terms except that which is necessary for a «reversal» cycle.

Karnaugh maps [6] are geometrical equivalents of binary systems of this type, and are frequently used to help explain the minimalization process. However, these maps are limited. A 4-by-4 map explains a 4-component system. As the size of the system increases by n, the size of the map required increases by 2^n.

The problem which limits the applicability of the method as developed by Quine, and these modifications of it, is that it is necessary to identify each and every element (n-vector) in the lattice. Thus, the size of the lattice which can be minimalized is too limited. We believe that the logical design departments of some computer manufacurers may well be at ware of methods far superior to those described in the open literature, but tend to keep this information as trade secrets. Hence, the best known techniques probably have not been published.

In a forthcoming paper for the *Notre Dame Journal of Formal Logic* [7], Locks developed a set-theoretic explanation of minimalization, including also an improved computerizable version of the method, which makes a full expansion of the polynomial unnecessary. This is achieved by incorporating a «reversal» technique, a form of perturbation applied whenever the polynomial is irredundant, that is, no term is a proper subset of any other term and no further reduction of the number of terms or indicators is visible. To test an irredundant form for minimality, reversal partitions the largest or first term into two complementary subsets. A mixing and matching then takes place to determine if it is possible to simplify by recombining the expanded terms with other subsets. At each reversal, one of three decisions results: the original irredundant form is also the unique minimal form, alternative minimal forms are obtained, or else reduction continues.

The earlier paper (i.e., the one still forthcoming) describes applications of the reversal process including examples taken from [1], but does not explicitly define the steps involved. In this paper, the sequence of steps and alternatives taken is described, somewhat in the form of a flow chart. We suspect there may be a number of good ways of achieving reversal, using heuristics or algorithms which remain yet to be discovered.

BASIC CONCEPTS

TERMINOLOGY

A system f, $f = 0, 1$, is a binary-valued function of n binary-valued indicators, $x_i = 0, 1$, $i = 1, \ldots, n$. The universal set u has 2^n elements (binary n-vectors), because for every component there are two choices; these include the unique zero element $(0, \ldots, 0)$ and the unique one element $(1, \ldots, 1)$. There are 2^{2^n} lattices (subsets) of u, because for every lattice, every element (n-vector) is either included or not included. At the extremes, we have the unique null (empty) set $\varnothing$, the lattice with no elements, and its complement, u.

Every lattice in u can be characterized as the «inclusive-or» union of a collection of complete subsets, also called *sublattices,* frequently in a variety of ways. A sublattice is identified by m fixed binary-valued indicators, $m \leqq n$, common to all elements in that set, with the remaining $n — m$ indicators all being free to assume either value, zero or one.

A complete subset includes both its g.1.b., with all free indicators zero, and its l.u.b., with all free indicators unity. There are other descriptions. A complete subset is a kind of free Boolean algebra (Halmos [8]), with a unique zero element, the g.l.b., a unique one element, the l.u.b., subject to component-by-component rules of Boolean addition $(0 + 0 = 0; 0 + 1 = 1 + 1 = 1)$, multiplication $(0 . 0 = 0 . 1 = 0; 1 . 1 = 1)$

and inversion ($\overline{1} = 0$; $\overline{0} = 1$) for the free components only, but not the fixed-valued ones.

In [9], Locks proves that there are 3^n complete subsets of u. This can be rationalized as follows. There are three choices for every one of the n components: fixed zero, fixed one, or free. Since a sublattice can be viewed as a set generated by the intersection of a subset of these components, there are 3^n ways of developing sublattices. The proof in Reference [9] is based on an actual count of the number of sublattices. The alternative rational explanation in this paragraph was suggested to the author by J. Chinal in a personal conversation.

The terms of a Boolean polynomial are its sublattices, that is, each term is the logical product or intersection of a subset of indicators representing fixed-valued components. The polynomial is the inclusive or union or join of these terms. A polynomial, however, is in general not a unique representation of a lattice. At one extreme is a full element-by-element listing, such as in a truth table, where every single element is accounted for. The other extreme, which is the desired one, is the minimalized form in which the lattice is described with the smallest number of terms (i.e., the largest possible subsets), and hence, also the smallest number of indicators.

THE MODIFIED QUINE ALGORITHM

The modified Quine algorithm for Boolean polynomial minimalization is described in a paper by Locks [7], to be published in the *Notre Dame Journal of Formal Logic*. It includes three different types of reduction, and the reversal test for uniqueness. That paper included examples of applications, taken mostly from the examples in [1]. This section summarizes that paper. In the sequel in the following section we present an algorithmic description of the reversal process, together with examples of how it works, adapted from the examples in Zissos and Duncan [10].

THE THREE TYPES OF MINIMALIZATION OPERATIONS

Minimalization includes three operations. For identification purposes, in the sequel we shall call them type-1, type-2, and type-3, respectively.

A type-1 operation is the deletion of a redundant sublattice, which is a proper subset of another set already in the lattice, for example

$$ab \vee a = a.$$

The set «a» includes «ab» as a proper subset. If the value is fixed, say $a = 1$, then any element with $a = 1$ is already a member of the set, regardless of the value of b.

A type-2 operation merges (joins ?) two complementary sublattices which differ only in the value of a single indicator, for example

$$abc \vee ab\bar{c} = ab.$$

This also says that the sublattice with $a = 1$ and $b = 1$ can be partitioned into two subsets, with $c = 1$ and $c = 0$, respectively. In the reversal method, when an irredundant form is obtained, to test it for uniqueness one term is partitioned into two subsets by reversing the type-2 operation, and these are recombined with other terms if it is possible to do so.

A type-3 operation is the elimination of a redundant indicator, for example

$$abc \vee a\bar{c} = ab \vee a\bar{c}.$$

This is proved as follows:

$$\begin{aligned} abc \vee a\bar{c} &= abc \vee ab\bar{c} \vee a\bar{b}\bar{c} \\ &= (abc \vee ab\bar{c}) \vee (ab\bar{c} \vee a\bar{b}\bar{c}) \\ &= ab \vee a\bar{c}. \end{aligned}$$

A type-3 operation reduces by one the number of indicators in a longer term for which there is a complementary sublattice which is a subset of a shorter term .This is done when the longer term has the same indicators as the shorter term, and the same values, except for one which is complemented. The complemented indicator is deleted in the longer term.

REVERSAL

Reversal is employed when one of the three types of minimalization can no longer be applied, to test by a search process for alternative ways of forming sublattices. It is initiated by expanding a term into two complementary subsets, and then trying to recombine these with other terms. Expansion is continued with other terms, if necessary, until a decision can be reached. There are three possible decisions:

1. The original form is the unique minimal form
2. There are alternative minimal forms
3. Minimalization can be continued.

In order to computerize reversal, it is necessary to develop an algorithm which can work efficiently on a wide variety of problems. A new general algorithm of this type is described in the following section. In this section, we show an example of reversal, in the context of a problem with a known answer, so that a smaller number of steps are required than would be needed in an algorithmic solution.

Since each reversal is initiated by expanding a single term into complementary subsets, it is better to do so first on the longer terms (smaller sets) than the shorter ones. Having more indicators than the shorter terms, there are more possibilities of recombining with other sets before the size of the polynomial becomes very large. Therefore, we recommend that the polynomial be rearranged so that the longer terms come first.

Example 1

This example is taken from [1], p. 528. For the polynomial

$$pqr \vee p\bar{r} \vee pq\bar{s} \vee \bar{p}r \vee \bar{p}\bar{q}\bar{r}\bar{s},$$

rearrange in order of the size of the term, as follows,

$$\bar{p}\bar{q}\bar{r}\bar{s} \vee pqr \vee pq\bar{s} \vee p\bar{r} \vee \bar{p}r.$$

No type-1 or type-2 operations can be performed, so that it is necessary to use only type-3 at the outset.

The first term combines with the fifth term to yield

$$\bar{p}\bar{q}\bar{r}\bar{s} \vee \bar{p}r = \bar{p}\bar{q}\bar{s} \vee \bar{p}r.$$

The second term combines with the fourth,

$$pqr \vee p\bar{r} = pq \vee p\bar{r}.$$

The polynomial thus simplifies to

$$\bar{p}\bar{q}\bar{s} \vee pq \vee pq\bar{s} \vee p\bar{r} \vee \bar{p}r.$$

A type-1 operation deletes the third term since

$$pq \vee pq\bar{s} = pq.$$

This yields one of the four alternative minimal forms

$$\bar{p}\bar{q}\bar{s} \vee pq \vee p\bar{r} \vee \bar{p}r. \tag{1}$$

It is not possible to simplify any further by using either a type-1, type-2, or type-3 operation. To obtain the other three minimal forms, reverse by first expanding the first term of Equation (1). This yields

$$\bar{p}\bar{q}\bar{r}\bar{s} \vee \bar{p}\bar{q}r\bar{s} \vee pq \vee p\bar{r} \vee \bar{p}r.$$

Since pqrs is a subset of pr it is deleted by a type-1 operation. Continue reversing by expanding pq to obtain

$$\bar{p}\bar{q}\bar{r}\bar{s} \vee pqr \vee p\bar{r} \vee \bar{p}r. \tag{2}$$

Note that pqr was deleted by a type-1 operation, since it is a subset of pr. By combining both the first and second terms of Equation (2) with the third by type-3 operations, we obtain the minimal form

$$\bar{q}\bar{r}\bar{s} \vee pq \vee p\bar{r} \vee \bar{p}r. \tag{3}$$

By combining both the first and second terms with the fourth, obtain

$$\bar{p}\bar{q}\bar{s} \vee qr \vee p\bar{r} \vee \bar{p}r. \qquad (4)$$

Finally, by combining the first term with the third and the second with the fourth, we have

$$\bar{q}\bar{r}\bar{s} \vee qr \vee p\bar{r} \vee \bar{p}r. \qquad (5)$$

Equations (1), (3), (4) and (5) are the four alternative minimal forms for this problem.

AN ALGORITHM FOR MINIMALIZATION AND REVERSAL

In the foregoing, an example was shown of a reversal-type of minimalization solution with a known answer. The number of reversal steps was considerably smaller than what would be needed if the result is not known, as is usually the case, and if one were to build a computer program to search for a solution. In this section we develop an algorithmic solution, including arranging terms, and the sequencing of operations and decisions in both the forward (minimalizing) and reversal modes.

This algorithm has not yet been computer tested. It seems to work adequately for hand methods, as is shown by the examples in the sequel.

SEQUENCING OPERATIONS IN THE FORWARD (MINIMALIZATION) MODE

The three types of operations are performed in the same sequence in which they are numbered, type-1 first, type-2 second and type-3 third. Since any type-2 or type-3 operation results in a larger set (shorter term) which one of the other terms could be a subset of, the cycle starts over again with a type-1 operation whenever all type-2 operations, or else all type-3 operations are completed. If no simplification is performed on a given cycle, the next step is a reversal.

ARRANGEMENT OF TERMS AND PAIRWISE MATCHINGS

Terms are arranged in order of size, longer terms (smaller sets) first. The more indicators a term has relative to the other terms of the polynomial, the more chances there are that it or its descendants will vanish by a type-1 operation. This can be particularly significant in reversal, because splitting the longer terms first cuts down the potential number of pairwise reverse-type-2 splittings, and therefore, also the size of the polynomial.

There could be some strategic points where it might be advisable to resequence the terms.

With terms arranged according to size, simplifications are the result of pairwise matchings front to back. That is, the first (longest) term is compared with the last (shortest) term first, then with the next to the last, second from the last, etc.; then the second term is compared with the last, next to the last, etc.; etc.

If all terms are of the same size, the terms with the most frequently used indicator should be bunched together at the beginning. It is very likely that this indicator would not vanish in further simplification, and increase the number of chances to form terms including this indicator, out of other sets.

REVERSAL SEQUENCE

When operating in the reversal mode, the number of indicators in some of the terms, and the number of terms, are both increasing. Since these increases result from reverse-type-2 operations, in order not to get the original polynomial back again, terms having the same parent are not recombined by a type-2 operation.

During reversal, there are both reversal operations (i.e., reverse-type-2) and forward operations. Forward type-1 operations are always performed in either the forward or reversal mode to keep the size of the polynomial as small as possible.

Forward type-2 operations are performed with terms having different parents only. A type-3 operation is not performed until after a forward type-2 of this kind. Either type-2 or type-3 can be performed at the very end of the reversal cycle after expanding the last term.

In reversing the term «ab» in ab $\vee$ d, the reversal steps are

$$\begin{aligned} & abc \vee ab\bar{c} \vee d \\ = \ & abc\bar{d} \vee ab\bar{c}\bar{d} \vee d. \end{aligned}$$

Since the maximum number of indicators have been accounted for, and since all terms are clearly partitioned (i.e., no two terms have any elements in common), we simplify in the forward mode to obtain

$$ab\bar{d} \vee d = ab \vee d.$$

EXAMPLES

Examples 2, 3, and 4 are all adapted from Zissos and Duncan [10], pp. 177-179. Explanations are included as needed. Since all additions in the sequel are logical, the arithmetic plus (+) sign is used in place of logical «or» ($\vee$).

Example 2. Minimalize $AB + \bar{A}C + \bar{B}D + \bar{C}D$

Since all terms have the same size and since D is the most frequently appearing fixed indicator, the polynomial is rearranged so that all terms including a D come at the beginning. Also note that no type-1, -2, or -3 operations are possible, so that it is necessary to reverse at the outset.

$$\begin{aligned}
& \bar{B}D + \bar{C}D + AB + \bar{A}C \\
= \ & A\bar{B}D + \bar{A}\bar{B}D + \bar{C}D + AB + \bar{A}C && \text{(reversal)} \\
= \ & A\bar{B}CD + \bar{A}\bar{B}CD + \bar{C}D + AB + \bar{A}C && \text{(reversal)} \\
= \ & A\bar{B}CD + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + A\bar{B}\bar{C}D + \bar{A}\bar{B}\bar{C}D \\
& \qquad + AB + \bar{A}C && \text{(reversal)} \\
= \ & A\bar{B}D + \bar{A}\bar{B}D + \bar{A}B\bar{C}D + AB + \bar{A}C && \text{(type-2)} \\
= \ & AD + \bar{A}\bar{B}D + \bar{A}BD + AB + \bar{A}C && \text{(type-3)} \\
= \ & AD + \bar{A}D + AB + \bar{A}C && \text{(type-2)} \\
= \ & D + AB + \bar{A}C && \text{(type-2)}
\end{aligned}$$

Note that the forward type-2 operation at the fourth step (combining terms from different parents) initiated breaking down the polynomial so that the remaining simplifications could be made.

Example 3

$$\begin{array}{lll} & ACD + \bar{A}BD + A\bar{B} + \bar{A}\bar{C} & \\ = & ABCD + \bar{A}BD + A\bar{B} + \bar{A}\bar{C} & \text{(reversal)} \\ = & ABCD + \bar{A}BCD + A\bar{B} + \bar{A}\bar{C} & \text{(reversal)} \\ = & BCD + A\bar{B} + \bar{A}\bar{C} & \text{(type-2)} \end{array}$$

Example 4

$$\begin{array}{lll} & AC + AB + \bar{A}C + \bar{A}\bar{B} & \\ = & ABC + AB + \bar{A}C + \bar{A}\bar{B} & \text{(reversal)} \\ = & ABC + ABC + AB\bar{C} + \bar{A}C + \bar{A}\bar{B} & \text{(reversal)} \\ = & A\bar{C} + ABC + \bar{A}C + \bar{A}\bar{B} & \text{(type-2)} \\ = & A\bar{C} + ABC + \bar{A}BC + \bar{A}\bar{B} & \text{(reversal)} \\ = & A\bar{C} + BC + \bar{A}\bar{B} & \text{(type-2)} \end{array}$$

The alternative minimal form is obtained by continuing the reversal

$$\begin{array}{lll} & AB\bar{C} + A\bar{B}\bar{C} + BC + \bar{A}\bar{B} & \text{(reversal)} \\ = & AB\bar{C} + A\bar{B}\bar{C} + ABC + \bar{A}BC + \bar{A}\bar{B} & \text{(reversal)} \\ = & AB + A\bar{B}\bar{C} + \bar{A}BC + \bar{A}\bar{B} & \text{(type-2)} \\ = & AB + A\bar{B}\bar{C} + \bar{A}BC + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} & \text{(reversal)} \\ = & AB + \bar{B}\bar{C} + \bar{A}C & \text{(type-2)} \end{array}$$

CONCLUSION

The method appears to give satisfactory results for a variety of problems. Further computer experimentation would be desirable.

Oklahoma State University Mitchell O. Locks

REFERENCES

[1] Quine, W.V. (1952) «The Problem of Simplifying Truth Functions,» *American Mathematical Monthly*, 59, pp. 521-531.

[2] McCluskey, E.J., Jr. (1956), «Minimization of Boolean Functions,» *Bell System Technical Journal*, v. 35, pp. 1417-1444.

[3] Chinal, J. (1973), *Design Methods for Digital Systems*, Springer-Verlag, New York, Heidelberg, Bonn, 1973. Translation of the French edition, *Techniques Booléenes et Calculateurs Arithmétiques*, Dunod, Paris, 1967.

[4] Phister, Montgomery, Jr. (1958), *Logical Design of Digital Computers* Wiley.

[5] Korfhage, Robert R. (1966), *Logic and Algorithms*, Wiley.

[6] Karnaugh, M. (1953), «The Map Method of Synthesis of Combinational Logic Circuits,» *Communications and Electronics*, No. 9, 593-599.

[7] Locks, M.O., (forthcoming) «Minimalization of Boolean Polynomials, Truth Functions and Lattices,» *Notre Dame Journal of Formal Logic*.

[8] Halmos, Paul R. (1963), *Lectures on Boolean Algebra*, D. van Nostrand.

[9] Locks, M.O., (forthcoming) «Logical and Probabilistic Analysis of Boolean Systems,» *Notre Dame Journal of Formal Logic*.

[10] Zissos, D. and F.G. Duncan (1973), «Boolean Minimization,» *Computer Journal*, 16, Number 2, pp. 174-179.