Enhancing Quine-McCluskey

Adrian Duşa
University of Bucharest
2007

Abstract

Currently, the only algorithm that yields an exact solution to the boolean minimization problem is the well-known Quine-McCluskey, but almost all software solutions employ different implementations because of its two fundamental weaknesses: it is memory hungry and slow for a large number of causal conditions.

This paper proposes an alternative to the classical Quine-McCluskey algorithm, one that addresses both problems, and especially the one of memory consumption. The solutions of this new algorithm are also exact, but they are produced not by following the cumbersome classical algorithm but using a more direct and faster approach.

Memory restrictions limit the number of input variables (causal conditions) at a ceiling of about 14 or 15 (because each new variable expands the memory usage in a geometric proportion), where this alternative uses only a very small fraction of memory and it can process about 20 input variables with acceptable speed.

1. The problem

Boolean minimization is a technique used in many scientific area, starting with computers and electric engineering. In the social sciences, it is used to finding the minimum combinations of causal conditions necessary to trigger the outcome. The analysis is called QCA (from Qualitative Comparative Analysis); it was introduced by Ragin (1987) and is currently under a strong development process, both theoretical and in terms of software implementations.

When studying a phenomenon, QCA researchers usually code the presence of a causal condition (or of the outcome) with 1 and the absence with 0, but it is also customary to code the presence with upper literals and the absence with lower literals. In this notation, “A” is the literal used to code the presence of the causal condition A, and “a” is the literal used to coding the absence of the same causal condition.

In set logics, “a” is the set of all elements falling outside of the set “A”, so that the reunion of the two sets (“a” AND “A”) cover the whole spectrum of possibilities (in this case the whole spectrum of possible states of a causal condition), and such a reunion is called TRUE and is equal to 1 (here the unity does not represent the presence of a causal condition but is used instead of the word “everything”).

The classical approach that finds the minimum combination of literals necessary to trigger an outcome is the Quine-McCluskey algorithm, developed by McCluskey (1956) but incorporating Quine's (1952, 1955) previous efforts. This approach starts from the unique combinations of presence/absence in the truth table and minimizes those pairs of combinations differing by only one literal (or bit, or state of causal condition).

If A, B and C are three causal conditions that affect the outcome E (or effect), this is a simple example of boolean minimization:

A	B	C	E
1	0	1	1
1	1	1	1
1	x	1	1

In both cases, the outcome is present, just like the causal conditions A and C. Only the causal condition B has two different states, being absent in the first case

and present in the second. Using Mill's (1843) inductive cannons, we can infer that causal condition B has no direct effect over the outcome.

The same conclusion is drawned using logic semantics: the two cases can be re-written as AbC + ABC. It is visible that AC is a common factor over the two cases, therefore the expression can be rewritten as AC(b + B).

As the reunion (here denoted by the plus “+” sign) of the set “b” and the set “B” is logically equal to 1, the result of this expression is AC, which is the same conclusion with AxC (where “x” is used to indicate the elimination of the causal condition B). AC is called a prime implicant, and it can be further minimized if different by only one literal with another prime implicant.

Boolean minimization is an iterative process that implies comparing every possible pair of input combinations, and in the next iterations every possible pair of prime implicants until nothing can be further minimized. The remaining prime implicants are then used to construct the so-called “prime implicants chart” in order to find the final solution.

For a handful of cases and a relatively small model, this technique can be easily followed by hand, but for a more complex model (with many causal conditions) and more input combinations entered in the analysis, the number of potential paired comparisons is extremely large and it grows in a geometrical proportion.

The analysis starts with the truth table, where all possible combinations of presence/absence of causal conditions are found. Table 1 presents such a table for three causal conditions.

TABLE 1 ABOUT HERE

The number of possible combinations is equal with the number of rows in the truth table and it can be computed with the formula 2^k, where k is the number of causal conditions. For a simpel case with three causal conditions, the truth table has eight rows but for a more complicated model with say 15 causal conditions the truth table has 2^{15} = 32,768 rows. This is not a very big number, but if 30,000 combinations would be included in the analysis, the number of potential paired comparisons is equal to almost 22,5 millions.

As 22,5 million comparisons usually yield many millions of prime implicants in the next iterations, the number of possible paired comparisons is rapidly growing to infinity. Even more, this has to be multiplied with the number of causal conditions to have a complete picture of the total number of comparisons being made. In our example, we compared three pairs: the states of the causal condition A in those two cases, the state of the causal condition B in those to cases and the same for the causal condition C. A minimization can only be made if only one such pair is different, therefore every one of them has to be compared. For 15 causal conditions, the real number of comparisons is therefore 22.5 million multiplied with 15.

An exact solution is obtained only by completing all iterations until nothing can be further minimized; the exact Quine-McCluskey algorithm is therefore bound to be slow for a large number of causal conditions, therefore this number has to be very limited; it usually cannot handle more than 10 or 11 causal conditions.

Efforts have been made (Dusa, 2007) to find alternative solutions. One approach is to treat every combination not as a series of 1s and 0s, but as a single line number in a 3^k matrix. There is a mathematical rule by which two such line numbers can be minimized in another line number from the same matrix, therefore the total number of comparisons is drastically reduced; the algorithm does not work with matrices anymore (a truth table is such a matrix) but with vectors of numbers. The memory needed to complete the algorithm is reduced by factors of tens, depending of the number of causal conditions in the model.

The number of paired comparisons can be further reduced to the strictly necessary because not all comparisons are minimizable; for example, if two combinations differ by more than one literal, they cannot be minimized. The solution for this problem is to produce a so-called “differences matrix”: there is a finite number of possible (minimizable) paired comparisons, and for each line in the 3^k matrix the minimizable pairs are found in the powers of 3 using the forward rule.

This approach yields exact boolean solutions for a number as large as 14 or even 15 causal conditions; however, it still suffers from the same fundamental problem: every new causal condition included in the analysis exponentially increases the number of possible paired comparisons. Cutting this number using mathematical formulae helps but eventually the algorithm is limited. Even though

it doesn't work with matrices that quickly fill up memory, vectors can also grow quickly when geometrically multiplied by 3; eventually, these vectors will certainly fill up the entire computer's memory.

A different algorithm is needed when using more than 15 causal conditions, one that would yield the same exact solutions as the classical Quine-McCluskey.

2. The solution

Once the last iteration has yielded the final prime implicants that cannot be further minimized, these are used to construct the prime implicants chart. This chart has the initial combinations included in the analysis (taken from the truth table) positioned on columns and the minimized prime implicants positions on rows. Table 2 presents such a chart, using the third, fourth, fifth and sixth initial combinations from Table1. These lines are the combinations “010”, “011”, “100“ and “101”, or expressed as a combination of literals they are: “aBc”, “aBC”, “Abc” and “AbC”.

TABLE 2 ABOUT HERE

When minimizing these combinations, the exact Quine-McCluskey algorithm has yielded three prime impicants: “01x” and “10x” or expressed as literals “aB” and “Ab”

The table present some “x” signs in the middle; these signs point the fact that a prime implicant explain an initial combination included in the analysis. For example, the prime implicant “aB” explains both “aBc” and “aBC”. Because “aB” can lead to both initial combinations, it is said that it is a superset of those combinations (or inversely “aBc” and “aBC” are subsets of “aB”).

The final solution is the one that contains the minimum number of prime implicants needed to cover all columns in the prime implicants chart. As both minimized prime implicants are essential, the final solution is: “aB + Ab”.

The solution proposed in this paper starts from two fundamental observations:

- the initial combinations are subsets of the prime implicants
- the combinations that were excluded from the analysis are **not** subsets of the minimized prime implicants.

The remaining combinations are: “000”, “001”, “110” and “111”, or expressed as literals: “abc”, “abC”, “ABc” and “ABC”.

Neither “aB” nor “Ab” can be found in these four excluded combinations, therefore they are not subsets of the minimized prime implicants.

These observations were also noted by Cronqvist (2002) when presenting the fundamentals of his Graph Based Agent implemented in the Tosmana (2007) software. The Graph Agent searches an optimum path in a multidimensional matrix in order to find the prime implicants that respect both these observations, but this approach is not guaranteed to be exact. Multiple test examples with 11, 12 and 13 causal conditions have shown that Tosmana yields different solutions from the exact Quine-McCluskey algorithm.

Instead, my solution follows an exhaustive calculation that is guaranteed to produce an exact solution by making extensive use of the ideas presented in the “Mathematical approach to the boolean minimization problem” (Dusa, 2007).

Following that approach, this new algorithm works with vectors of line numbers instead of (multidimensional) complex matrices. There is a finite number of both prime implicants and initial combinations, which are all rows in a 3^k matrix (where k is the number of causal conditions). These combinations were first noted by Ragin in his well known “Fuzzy-set social science” (2000), where he called them “logically possible groupings”. Table 3 presents a 3^k matrix for three causal conditions.

TABLE 3 ABOUT HERE

In this matrix, a different notation is used: the presence of a causal condition is coded with 2, the absence of a causal condition is coded with 1 and a minimized causal condition is coded with 0. It is a 3^k matrix not because there are three causal conditions but because of the three codes used: 0, 1 and 2. The 3^k matrix is best described as containing all possible combinations in base 3; there are such matrices for two, three, four and generally for any number of causal conditions.

Using line numbers only, the initial combinations can be found in the lines 17, 18, 23 and 24. Similarly, the minimized prime implicants can be found on lines 16 and 22. There is another way to obtain these two line numbers corresponding to the prime implicants, bypassing the classical minimization process.

First, any initial combination of causal conditions can be split in supersets, and the number of possible supersets that can be obtained is (as usual) finite. The formula to compute this number is:

[1] $$\sum_{k=1}^{n} \binom{n}{k} = 2^k - 1$$

where

[2] $$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}$$

Formula [2] gives the combinations of *k* causal conditions taken from *n* possible ones, using factorial notation.
It sometimes has different notations, such as: $_nC_k$ or C_n^k

For example, the combination “aBc” has the following $2^3 - 1 = 7$ supersets:
“a”, “B”, “c”
“aB”, “ac”, “Bc” and
“aBc”

Counting out the input combination itself, the actual number of prime implicants is $2^k - 2$.

There are 3 combinations of a single literal, three combinations of two literals and one combination involving all three literals, seven in total. In set theory, a set is both its superset and its subset: “aBc” is a superset of the same combination “aBc”, and inversely it can be thought of its own subset.

The corresponding line numbers in the 3^k matrix (in the same order) are:
10, 7, 2
16, 11, 8 and
17

These numbers can be computed following the same technique presented in the “Mathematical approach”: using the powers of 3. The elements in the first column will be multiplied by $3^2 = 9$, the elements in the second column will be multiplied by $3^1 = 3$ and the elements in the third column will be multiplied by $3^0 = 1$. The multiplication vector is therefore 9, 3 and 1.

For example, the initial combination “101” from the truth table is the equivalent of the combination “**1 2 1**” in the 3^k matrix (line 17).

Multiplying with the powers of 3 vector gives:

$1 \cdot 3^2$, $2 \cdot 3^1$ and $1 \cdot 3^0$ which is the combined vector: **9**, **6** and **1**.

It should be noted that the combined vector is computed in base 10 that starts with 0; as line numbers start with 1, each computation should be increased by 1 in order to obtain the required line numbers.

The rest of the calculation is straighforward:

9 + 1 =	10	
6 + 1 =	7	
1 + 1 =	2	are the first three line numbers
9 + **6** + 1 =	16	
9 + **1** + 1 =	11	
6 + **1** + 1 =	8	are the next three line numbers, and
9 + **6** + **1** + 1 =	17	is the initial line number.

It all boils down to computing a vector of powers of 3 and use that to multiply each initial combination of causal conditions included in the analysis. In this example, the four initial combinations are found in the lines 17, 18, 23 and 24 (or combinations “121”, “122”, “211” and “212”).

Multiplying every combination of *k* causal conditions in each of these four initial line numbers yields the following unique supersets (or prime implicants):

2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 16, 17, 18, 19, 20, 21, 22, 23 and 24.

This first vector was obtained from the combinations to be explained by the analysis, usually associated with the presence of the outcome (but other combinations can be explained as well); these are the combinations used to construct the prime implicants chart, therefore I will call it the “explained *vector*” (from here on denoted by EXP).

Similarly, the resulting supersets (or prime implicants) obtained from the excluded combinations (which are found in the lines 14, 15, 26 and 27 in the 3^k matrix) are:

2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 19, 20, 21, 25, 26, 27

This second vector was obtained from the combinations specifically excluded from the analysis, usually associated with the absence of the outcome (but other

combinations can be excluded as well), therefore I will call it the "*excluded vector*" (from here on denoted by EXC)

The only prime implicants that are in the explained vector and **not** in the excluded one are found on the line numbers: 16, 17, 18, 22, 23, 24. I will call this the "*reduced explained vector*" (from here on denoted by REV)

Four of these line numbers are the initial combinations included in the analysis: 17, 18, 23 and 24. The fact that they were selected as possible prime implicants is a positive proof that this method is complete and exact. Sometimes, the initial combinations cannot be minimized at all, in which case they are equal to their own prime implicants and the solution is the same as the input combinations.

In this example, the line numbers are indeed minimizable, therefore the initial combinations are redundant. Section 4 of this paper presents a mathematical rule to find the redundant prime implicants, leaving only the minimum ones. The final solution in the example, after removing the initial four combinations, is found in the line numbers: 16 and 22.

In the 3^k matrix, these line numbers correspond to the base 3 combinations: "120" and "210".

Their correspondents in base 2 are: "01x" and "10x"

Translated in literals, these are: "aB" and "Ab", which are the *exact* prime implicants found in Table 2.

For a small number of causal conditions the solution presented is easy, but the algorithm quickly grows in complexity once this number is increased. Fortunately, the sets of positive and negative cases are always small, because most of the possible causal combinations do not have an association with the outcome.

3. Dealing with the remainders

For a large number of causal conditions, the total number of combinations of causal conditions is 2^k. These are possible causal patterns that can be observed when studying a phenomenon. Some of these causal combinations are associated with the presence, and the others with the absence of that phenomenon.

The two sets of combinations are the basis for the two sets of line numbers described in the previous section.

Ideally, the researcher should have positive information about each and every causal combination. However, there are two big impediments that hinder this

ideal situation:

- the social phenomena do not present all possible combinations of causal conditions, situation described in QCA as “limited diversity”
- QCA is a special analysis between the quantitative and qualitative worlds, working not with thousands of cases as in the quantitative world but with a handful of cases as in the qualitative world. It is therefore logically impossible to observe every possible combination with only a handful of cases.

Even in the quantitative approach, the situation would be the same: these studies usually employ samples of about 1,000 cases, but for 15 causal conditions there are 32,768 logically possible groupings. As some cases are similar due to the limited diversity, many possible groupings will remain without any information in respect to the outcome. These are the so-called “remainders” and they play a crucial role in the minimization process.

These are also the equivalent of the so-called “Don't care”s from the electrical engineering field, where 1 represent an open gate and 0 a closed gate. In this field, boolean minimization doesn't care if the gates from the remainders are open or not, as long as a more minimum solution is obtained.

The same interpretation is found in the social sciences: even if the outcome for these remainders is not known, the researchers include the remainders in the minimization process *assuming* the outcome is present (or positive). This paper does not discuss this assumption, it only offers solutions for the situation when these remainders are included in (or excluded from) the analysis. Table 4 present a hypothetical truth table that contains positive cases, negative cases and remainders.

TABLE 4 ABOUT HERE

The remainders are coded with the question mark “?”. They are responsible for the exponential complexity of the Quine-McCluskey algorithm; when including them in the minimization process, the total number of paired comparisons quickly explodes to infinity.

The solution proposed in this paper solves this problem, because finding the positive minimum prime implicants does not involve the remainders at all: this

approach uses only the two sets of positive and negative cases to find the positive and negative prime implicants, and select only those positive prime implicants that are not in the set of negative ones

Including the prime implicants in the analysis has absolutely no effect over the complexity of the algorithm, because the prime implicants chart is contructed using only the (very small) set of the explained combinations.

There is only one possible situation where the remainders are potentially influent: the case when they are specifically *excluded* from the analysis. In this situation, the algorithm should indeed take them into account in the EXC (excluded) vector and the complexity explodes.

When excluding the remainders, there is no other simpler and faster solution but to follow the classical Quine-McCluskey algorithm: without the remainders, the number of combinatons to be explained is usually very small and the classical algorithm easily finds a solution for any number of causal conditions.

This situation is exactly the inverse of the classical Quine-McCluskey algorithm: if there the complexity explodes when *including* the remainders, here it explodes when *excluding* them.

4. Finding the most parsimonious prime implicants

For a large number of causal conditions, the number of possible prime implicants grows very quickly, function of the combinations explained and the combinations excluded. There is no exact formula to compute this number, as some prime implicants are common for multiple initial combinations, but it can easily reach hundreds of thousands for more than 10 causal conditions.

REV is computed using an exhaustive method, but many prime implicants are not parsimonious as they are already contained by other ones. For example, if REV contains both the combinations “aB” and “aBc”, the last one is not parsimonious because it is a subset of the first. “aB” is a more minimum expression, therefore “aBc” has to be eliminated.

When all possible eliminations are performed, REV will contain only the most parsimonious expressions. Using the language of set theory, a parsimonious expression is a superset that has no other subsets in the same vector. Conversely, any expression (prime implicant) that has supersets in REV has to be eliminated.

Considering any prime implicant as a line number in the 3^k matrix, its supersets are always smaller line numbers and its subsets are always bigger line numbers. For example, if the prime implicant of interest is “Bc” (line number 8 in Table 3, or combination “0 2 1”), it has two supersets: “B” in the line number 7 (combination “0 2 0”) and “c” in the line number 2 (combination “0 0 1”). Its subsets are “aBc” in the line number 17 (combination “1 2 1”) and “ABc” in the line number 26 (combination “2 2 1”).

As REV contains line numbers, the first step is to sort it in ascending order, so that subsets (prime implicants that are not parsimonious) will always be found on the right (bigger numbers).

There are two possible strategies to identify the most parsimonious prime implicants:

- from right to left: starting from the biggest number, if it has supersets in REV it should be eliminated, if not it should be kept and so on
- from left to right: starting from the smallest number, eliminate all its subsets from REV, continue with the second surviving number and so on.

The first strategy is the most expensive in terms of computer cicles, as it checkes all prime implicants. The second strategy is the most efficient and faster, leaving only one problem to solve: how to find the subsets for a given minimum expression.

The solution is again found using the ideas presented in the “Mathematical solution”. For three causal variables, the powers of three are: 3^2, 3^1 and 3^0 (or simply 9, 3 and 1)

The following illustrative example uses a hypothetical REV, sorted in ascending order: (4, 6, 7, 9, 13, 15, 16, 19)

As minimum prime implicants are always among the first (smaller) numbers, the algorithm starts with line 4, which in Table 3 is the combination “0 1 0” or “b”, and executes the folowing steps:

- find which elements in the base 3 combination are equal to zero: in this case, the first and the third
- starting with the rightmost element (the third), progressively add $3^0 = 1$ two times: from 4 it advances to 5 and from 5 it advances to 6, resulting in the vector: (4, 5, 6); line 5 is “bc” and line 6 is “bC”, both subsets of line 4.

– the next element equal to zero is the first, and the algorithm adds 3^2 =9 two times to the previous vector: from (4, 5, 6) it advances to (13, 14, 15) and from (13, 14, 15) it advances to (22, 23, 24), resulting in the vector: (4, 5, 6, 13, 14, 15, 22, 23, 24)

The first number is the starting superset and it should not be eliminated, which leaves the final vector of subsets: (5, 6, 13, 14, 15, 22, 23, 24)

Among the initial vector (4, 6, 7, 9, 13, 15, 16, 19) the subsets of 4 are line numbers 6, 13 and 15; they are eliminated and the algorithm continues with the remaining vector: (4, 7, 9, 16, 19)

No other subsets of 4 exist, therefore the next superset to be verified is line number 7 (combination “0 2 0” or the “B” literal expression). Following the same procedure as above, its subsets are: (8, 9, 16, 17, 18, 25, 26, 27)

Among the remaining vector (4, 7, 9, 16, 19), two other prime implicants are not parsimonious: 9 and 16; they are eliminated and the surviving vector of the most parsimonious prime implicants is: (4, 7, 19)

This elimination approach is very fast because REV is shortened progressively, reaching its surviving form in the smallest possible number of iterations. In the end it contains the minimum number of the most parsimonious prime implicants that are further used to construct the prime implicants chart (which has another routine to further remove other redundant prime implicants).

5. Conclusions

In this paper I discuss the inherent problems related to the classical Quine-McCluskey algorithm and show its weaknesses as well as its strengths.

When the number of combinations included for minimization is (very) large, the classical algorithm becomes extremely slow and it eventually runs out of memory. For this situation I have presented another (exhaustive) procedure that produces exact solutions. In the opposite situation, when the number of excluded combinations is (very) large, this different approach becomes extremely slow due to the computation of the EXC (excluded) vector.

The algorithm presented in this paper *enhances* and extends the classical Quine-McCluskey: they help and complement each other, function of the situation when the remainders are included or excluded from the analysis.

References

Cronqvist, Lasse (2002): How MVQCA works. A short Introduction to the Ideas of the Algorithm used in TOSMANA. Powerpoint-Presentation retrieved from http://www.tosmana.net in September 2007

Cronqvist, L. (2007). Tool for Small-N Analysis [Version 1.3]. Marburg. Retrieved from http://www.tosmana.net in September 2007.

Dusa, A. (2007). A mathematical approach to the boolean minimization problem. Retrieved from www.compasss.org, Working Papers section in September 2007.

McCluskey, E.J. (1956). Minimization of Boolean Functions. The Bell System Techical Journal, November.

Mill, J.S. (1843). A System of Logic. 8th edition, reprinted in 1925. Longman, Green, and Co.: London. Retrieved from http://www.la.utexas.edu/cuws/index.html in January 2007.

Quine, W.V. (1952). “The Problem of Simplifying Truth Functions”. The American Mathematical Monthly, Vol. 59, No. 8.

Quine, W.V. (1955). “A Way to Simplify Truth Functions”. The American Mathematical Monthly, Vol. 62, No. 9.

Ragin, C.C. (1987). The Comparative Method. Moving Beyond Qualitative and Quantitative Strategies. University of California Press, Berkeley and Los Angeles, California.

Ragin, C.C. (2000). Fuzzy set social science. The University of Chicago Press.

Table 1. The truth table for three causal conditions

Line	A	B	C	Literals
1	0	0	0	abc
2	0	0	1	abC
3	0	1	0	aBc
4	0	1	1	aBC
5	1	0	0	Abc
6	1	0	1	AbC
7	1	1	0	ABc
8	1	1	1	ABC

Table 2. An example of a prime implicants chart

	aBc	aBC	Abc	AbC
aB	x	x	–	–
Ab	–	–	x	x

Table 3. The 3^k space for three causal conditions

Line	A	B	C	Literals
1	0	0	0	
2	0	0	1	c
3	0	0	2	C
4	0	1	0	b
5	0	1	1	bc
6	0	1	2	bC
7	0	2	0	B
8	0	2	1	Bc
9	0	2	2	BC
10	1	0	0	a
11	1	0	1	ac
12	1	0	2	aC
13	1	1	0	ab
14	1	1	1	abc
15	1	1	2	abC
16	1	2	0	aB
17	1	2	1	aBc
18	1	2	2	aBC
19	2	0	0	A
20	2	0	1	Ac
21	2	0	2	AC
22	2	1	0	Ab
23	2	1	1	Abc
24	2	1	2	AbC
25	2	2	0	AB
26	2	2	1	ABc
27	2	2	2	ABC

Table 4. A hypothetical truth table containing remainders

Line	A	B	C	Outcome
1	0	0	0	?
2	0	0	1	1
3	0	1	0	1
4	0	1	1	?
5	1	0	0	?
6	1	0	1	0
7	1	1	0	0
8	1	1	1	?