QUINE–MCCLUSKEY ALGORITHM

The Quine–McCluskey algorithm (or the method of prime implicants) is a method used for minimization of boolean functions which was developed by W.V. Quine and Edward J. McCluskey. It is functionally identical to Karnaugh mapping, but the tabular form makes it more efficient for use in computer algorithms, and it also gives a deterministic way to check that the minimal form of a Boolean function has been reached. It is sometimes referred to as the tabulation method.

The method involves two steps:

1. Finding all prime implicants of the function.
2. Use those prime implicants in a *prime implicant chart* to find the essential prime implicants of the function, as well as other prime implicants that are necessary to cover the function.

COMPLEXITY

Although more practical than Karnaugh mapping when dealing with more than four variables, the Quine–McCluskey algorithm also has a limited range of use since the problem it solves is NP-hard: the runtime of the Quine–McCluskey algorithm grows exponentially with the number of variables. It can be shown that for a function of n variables the upper bound on the number of prime implicants is $3^n/n$. If $n = 32$ there may be over $6.5 * 10^{15}$ prime implicants. Functions with a large number of variables have to be minimized with potentially non-optimal heuristic methods, of which the Espresso heuristic logic minimizer is the de-facto world standard.[1]

EXAMPLE:

STEP 1: FINDING PRIME IMPLICANTS

Minimizing an arbitrary function:

	A B C D	f
m0	0 0 0 0	0
m1	0 0 0 1	0
m2	0 0 1 0	0
m3	0 0 1 1	0
m4	0 1 0 0	1
m5	0 1 0 1	0
m6	0 1 1 0	0
m7	0 1 1 1	0
m8	1 0 0 0	1
m9	1 0 0 1	x
m10	1 0 1 0	1
m11	1 0 1 1	1
m12	1 1 0 0	1
m13	1 1 0 1	0
m14	1 1 1 0	x
m15	1 1 1 1	1

One can easily form the canonical sum of products expression from this table, simply by summing the minterms (leaving out don't-care terms) where the function evaluates to one:

$$f_{A,B,C,D} = A'BC'D' + AB'C'D' + AB'CD' + AB'CD + ABC'D' + ABCD.$$

Of course, that's certainly not minimal. So to optimize, all minterms that evaluate to one are first placed in a minterm table. Don't-care terms are also added into this table, so they can be combined with minterms:

```
Number of 1s  Minterm  Binary Representation
--------------------------------------------
1             m4       0100
              m8       1000
--------------------------------------------
```

2	m9	1001
	m10	1010
	m12	1100
3	m11	1011
	m14	1110
4	m15	1111

At this point, one can start combining minterms with other minterms. If two terms vary by only a single digit changing, that digit can be replaced with a dash indicating that the digit doesn't matter. Terms that can't be combined any more are marked with a "*". When going from Size 2 to Size 4, treat '-' as a third bit value. Ex: -110 and -100 or -11- can be combined, but not -110 and 011-. (Trick: Match up the '-' first.)

Number of 1s	Minterm	0-Cube	Size 2 Implicants	Size 4 Implicants
1	m4	0100	m(4,12) -100*	m(8,9,10,11) 10--*
	m8	1000	m(8,9) 100-	m(8,10,12,14) 1--0*
			m(8,10) 10-0	
2	m9	1001	m(8,12) 1-00	m(10,11,14,15) 1-1-*
	m10	1010		
	m12	1100	m(9,11) 10-1	
			m(10,11) 101-	
3	m11	1011	m(10,14) 1-10	
	m14	1110	m(12,14) 11-0	
4	m15	1111	m(11,15) 1-11	
			m(14,15) 111-	

Note: In this example, none of the terms in the size 4 implicates table can be combined any further. Be aware that this processing should be continued otherwise (size 8 etc).

STEP 2: PRIME IMPLICANT CHART

None of the terms can be combined any further than this, so at this point we construct an essential prime applicant table. Along the side goes the prime applicants that have just been generated, and along the top go the minterms specified earlier. The don't care terms are not placed on top - they are omitted from this section because they are not necessary inputs.

	4	8	10	11	12	15	
m(4,12)*	X				X		-100
m(8,9,10,11)		X	X	X			10--
m(8,10,12,14)		X	X		X		1--0
m(10,11,14,15)*			X	X		X	1-1-

Here, each of the *essential* prime applicants has been starred - the second prime applicant can be 'covered' by the third and fourth, and the third prime applicant can be 'covered' by the second and first, and is thus neither an essential. If a prime applicant is essential then, as would be expected, it is necessary to include it in the minimized Boolean equation. In some cases, the essential prime applicants do not cover all minterms, in which case additional procedures for chart reduction can be employed. The simplest "additional procedure" is trial and error, but a more systematic way is Patrick's Method. In the current example, the essential prime implicants do not handle all of the minterms, so, in this case, one can combine the essential implicants with one of the two non-essential ones to yield one of these two equations:

Both of those final equations are functionally equivalent to this original (very area-expensive) equation:

- Note: Refer to the book by P.S Manoharan titled "Digital Principles and System Design" for more detailed and easier explanation.