12.12 Quine-McCluskey

Basic algorithm

Quine-McCluskey is an algorithm for two-level logic optimization, suitable for computer automation due to using a tabular method (rather than graphical method like K-maps). Given a function's minterms, the algorithm's steps are:

1. *Generate PI's:* Create a table of minterms, then pairwise check minterms for i(j + j') opportunities, combining into new terms in a new column, repeating with new terms until no more combinations can be made. Each term that wasn't combined with another (minterms or new terms) is a prime implicant (PI).
2. *Find essentials:* Draw a table with PI's as rows and minterms as columns, putting a mark to indicate a PI covers a minterm. For any column with only one mark, the PI for that row is essential so is added to the cover. All minterms covered by that PI are also checked off as covered.
3. *Cover remaining:* Select minimal unadded prime implicants to cover remaining minterms.

PARTICIPATION ACTIVITY 12.12.1: Quine-McCluskey algorithm.

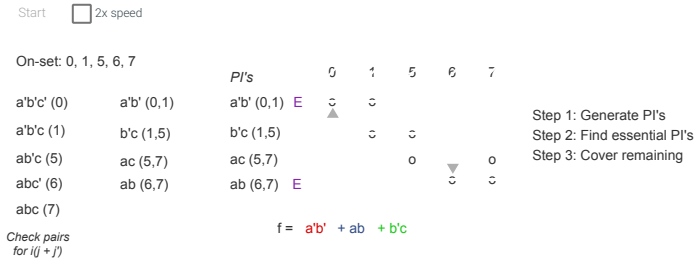
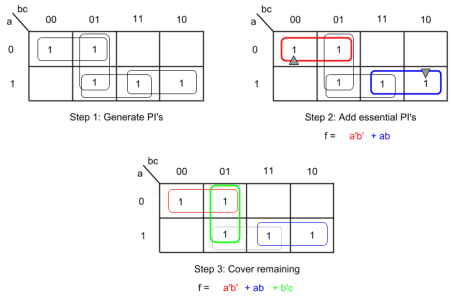
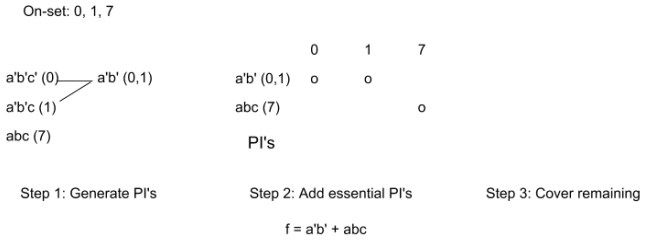


Figure 12.12.1: Equivalent K-map operations for the above Quine-McCluskey steps.



PARTICIPATION ACTIVITY 12.12.2: Quine-McCluskey: Simple 3-variable example.

Consider the partially-completed Quine-McCluskey algorithm tables below.



1) If all minterm pairs were checked for an i(j + j') opportunity, how many checks were made?
   - 1
   - 2
   - 3

2) How many checks for i(j + j') opportunities are done with a'b'?
   - 0
   - 1

3) How many PI's were generated?

1
2

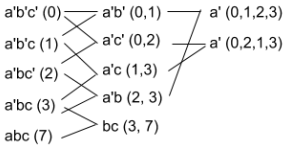
4) How many of the PI's are essential?
   - 1
   - 2

5) How many of the PI's should be added in Step 3 to complete the cover?
   - 0
   - 1

PARTICIPATION ACTIVITY 12.12.3: Quine-McCluskey: Another 3-variable example.

Consider the partially-completed Quine-McCluskey algorithm tables below.

On-set: 0, 1, 2, 3, 7



Step 1: Generate PI's

PI's	0	1	2	3	7
a' (0,1,2,3)	o	o	o	o	
bc (3,7)				o	o

Step 2: Add essential PI's

1) If all minterm pairs were checked for an i(j + j') opportunity, how many checks were made?
   - 5
   - 10
   - 25

2) How many two-literal prime implicants are generated?
   - 1
   - 2
   - 5

3) The left table lists three implicants that aren't combined with another: bc, a', and a'. Why does the right table only list two PI's?
   - Mistake
   - a' and a' are the same

4) How many PI's are essential?
   - 1
   - 2

5) Is Step 3 necessary?
   - No
   - Yes

Grouping terms by number of uncomplemented literals

Note: The above algorithm checked all minterm pairs (and then all new term pairs in the next column, etc.) for an i(j + j') opportunity. However, for an i(j + j') opportunity to exist, the terms must differ by exactly one literal. As such, an improvement to the algorithm's efficiency is to only check pairs whose number of uncomplemented literals differs by one. Ex: a'b'c has one uncomplemented literal (c) while abc has three (a, b, c). Because the difference is not one, checking that pair can be skipped. Thus, for a given column, the Quine-McCluskey algorithm first sorts terms into a group with zero uncomplemented literals, a group with one, a group with two, etc. Then, for a given term, the algorithm only compares with terms in the next group.

Petrick's method

Note: Step 3 requires an algorithm itself to find the minimum cover. A straightforward approach is called Petrick's method. The method is beyond this material's scope.

Exploring further:

- Online Quine-McCluskey minimizer (T. Thormaehlen)
- Article on Quine-McCluskey (embedded.com)
- Petrick's Method (Wikipedia)

Provide feedback on this section