On Quine-McCluskey Method

- Goal: find a minimum SOP form
- Why We Need to Find all PIs?

$$f(w,x,y,z) = x'y' + wxy + x'yz' + wy'z$$
$$= x'y' + x'z' + wxy + wy'z$$
$$= x'y' + x'z' + wxy + wxz$$
$$= x'y' + x'z' + wxz + wyz'$$

1. ***Are all terms PIs?***
2. ***Is the form optimal?***
3. ***Is the form unique?***

- How We Find Them?
  - Quine's tabular: start with minterm, the smallest I
  - Iterated consensus: complete sum theorem 4.5.1
  - Recursive: complete sum theorem 4.6.1

Quine-McCluskey Method

Problem: Given a Boolean function *f* (may be incomplete), find a minimum cost SOP formula.

# of literals

Q-M Procedure:

1. Generate all the PIs of *f*, $\{P_j\}$
2. Generate all the minterms of *f*, $\{m_i\}$
3. Build the Boolean constraint matrix B, where B_{ij} is 1 if $m_i \in P_j$ and is 0 otherwise
4. Solve the minimum column covering problem for B

Example: Quine-McCluskey Method

$$f(w,x,y,z) = x'y' + wxy + x'yz' + wy'z$$

	wxy	wxz	wyz'	$wy'z$	$x'y'$	$x'z'$
$wx'y'z'$					1	1
$w'x'y'z$					1	
$w'x'y'z'$					1	1
$wxyz$	1	1				
$wxyz'$	1		1			
$wx'yz'$			1			1
$w'x'yz'$						1
$wxy'z$		1		1		
$wx'y'z$				1	1	

minimum cover(s):

$\{x'y', x'z', wxy, wxz\}$,

$\{x'y', x'z', wxy, wy'z\}$,

$\{x'y', x'z', wxz, wyz'\}$.

Two-Level Logic Synthesis
-- Unate Covering Problem

Unate and Binate

- A function $f(x_1,\cdots x_i\cdots x_n)$ is positive unate in x_i if its cofactor f_{x_i} includes $f_{x_i'}$

  Negative unate is defined in a similar way. If a function is neither positive unate nor negative unate in a variable, it is called binate in this variable.

  A function is positive/negative unate if it is so for all variables, otherwise it is called binate.

- Example: $f(x,y,z) = xy + xz' + yz'$
  - f is positive unate in x: $f_x=y+z'+yz'$, $f_{x'}=yz'$
  - f is positive unate in y: $f_y=x+xz'+z'$, $f_{y'}=xz'$
  - f is negative unate in z: $f_z=xy$, $f_{z'}=xy+x+y$

Unate Covering Problem (UCP)

> Let M_{mxn} be a Boolean matrix (like the constraint matrix in Q-M), the UCP is to find a minimum number of columns to cover M in the sense that any row with a 1-entry has at least one of its 1-entries covered by these columns.

	wxy	*wxz*	*wyz'*	*wy'z*	*x'y'*	*x'z'*
wx'y'z'					1	1
w'x'y'z					1	
w'x'y'z'					1	1
wxyz	1	1				
wxyz'	1		1			
wx'yz'			1			1
w'x'yz'						1
wxy'z		1		1		
wx'y'z				1	1	

Solutions to UCP:

{x'y', x'z',wxy, wxz},

{x'y', x'z',wxy, wy'z},

{x'y', x'z',wxz, wyz'}.

Reduction Techniques

1. Check for essential columns and remove them;
2. Check for row dominance and remove all dominating rows;
3. Check for column dominance and remove all dominated columns;
4. Repeat 1, 2, 3 if there is any removal occurs.

What is left?

= If no rows/columns left, we find an optimal solution;

= Otherwise, this UCP instance is called cyclic.

Essential Columns

> A column is essential if it covers one 1-entry that cannot be covered by any other columns.

	wxy	wxz	wyz'	$wy'z$	$x'y'$	$x'z'$
$wx'y'z'$					1	1
$w'x'y'z$					1	
$w'x'y'z'$					1	1
$wxyz$	1	1				
$wxyz'$	1		1			
$wx'yz'$			1			1
$w'x'yz'$						1
$wxy'z$		1		1		
$wx'y'z$				1	1	

Column $x'y'$ is essential because of row $w'x'y'z$.

Essential columns can be removed because any cover must include them.

Row Dominance

> Row r_i dominates row r_j if r_i has all the 1-entries in r_j. r_i is dominating and r_j is dominated.

	wxy	*wxz*	*wyz'*	*wy'z*	*x'y'*	*x'z'*
wx'y'z'					1	1
w'x'y'z					1	
w'x'y'z'					1	1
wxyz	1	1				
wxyz'	1		1			
wx'yz'			1			1
w'x'yz'						1
wxy'z		1		1		
wx'y'z				1	1	

Row *wx'y'z'* dominates rows *w'x'y'z* and *w'x'yz'*

Dominating rows can be removed because that whenever one of their dominated rows is covered, they are covered as well.

Column Dominance

> Column p_i **dominates** p_j if p_i has all the 1-entries in p_j and p_i costs (e.g., number of literals) no more than p_j.

	1	2	3	4
1	1	1		
2	1	1		
3		1	1	1
4		1	1	1

Suppose all columns have the same cost, then column 2 dominates 1, 3, and 4. Columns 3 and 4 dominate each other.

Dominated columns can be removed because that their dominating columns cover all their 1-entries with no more cost. (optimal solutions may be lost, but at least one is guaranteed.)

Example: Reduction Techniques

$$\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 1 & 1 & & & & 1 \\ 2 & 1 & 1 & & & & \\ 3 & & 1 & 1 & & & \\ 4 & & 1 & 1 & 1 & & \\ 5 & & & & 1 & 1 & \\ 6 & & & & 1 & 1 & 1 \end{array}$$

1. No essential columns;
2. Row 1,4, and 6 dominates 2, 3, and 5 respectively;
3. Column 2 dominates 1 and 3, columns 4 and 5 dominate each other;
4. Now both columns become essential.

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 2 & 1 & 1 & & & \\ 3 & & 1 & 1 & & \\ 5 & & & & 1 & 1 \end{array} \Rightarrow \begin{array}{c|cc} & 2 & 4 \\ \hline 2 & 1 & \\ 3 & 1 & \\ 5 & & 1 \end{array} \Rightarrow \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 1 & 1 & & & & 1 \\ 2 & 1 & 1 & & & & \\ 3 & & 1 & 1 & & & \\ 4 & & 1 & 1 & 1 & & \\ 5 & & & & 1 & 1 & \\ 6 & & & & 1 & 1 & 1 \end{array}$$

Example: Cyclic UCP

No essential columns;

No row dominance;

No column dominance;

- Reduction technique stops.
- Normally multiple solutions exist in such case.
- How to find one optimal?
  - Implicit enumeration (exhaustive search)
  - Branch and bound

$$
\begin{array}{c}
\begin{array}{cccccc} \;1 & \;2 & \;3 & \;4 & \;5 & \;6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}
\begin{pmatrix}
1 & & & & & 1 \\
1 & 1 & & & & \\
 & 1 & 1 & & & \\
 & & 1 & 1 & & \\
 & & & 1 & 1 & \\
 & & & & 1 & 1
\end{pmatrix}
\end{array}
$$

Branch and Bound

Exhaustive search directed by easily computable bounds.

- Approach:
  1. Generate simpler instances of the same problem;
  2. Compute bound for each simpler instances;
  3. if there is direct solution better than all the bounds, stop;
  4. else go to step 1 for the instance with best bound;

- Example: traveling salesman

Branch and Bound Algorithm

- How to split?
  - One particular column is selected or not.
- What is a lower bound?
  - Graph version and MIS
- Why do we need an upper bound?
  - Early truncation of branches in the decision tree
- What can be an upper bound?
  - All columns
  - The current best solution

MIS: Lower Bound for UCP

$$
\begin{array}{c}
1\\ 2\\ 3\\ 4\\ 5\\ 6\\ 7
\end{array}
\begin{pmatrix}
1 & & & & 1 & \\
1 & 1 & & 1 & & \\
 & 1 & 1 & & & \\
 & & & 1 & 1 & 1\\
 & & 1 & 1 & & \\
 & 1 & & & & 1\\
1 & & 1 & & &
\end{pmatrix}
$$

Convert the constraint matrix to graph:

- Rows → nodes
- If two rows have one common 1-entry → an edge

UCP and MIS:

- A pair of unconnected nodes
- An independent set
- A maximal independent set
- The number of columns in a solution to UCP cannot be less than the size of a MIS.

MIS_QUICK: a heuristic to find a MIS

Pseudo-Code: Branch and Bound

1. Apply reduction techniques until they stop;
2. Calculate the lower/upper bounds;
3. Pick one column and split into two branches;
4. Check one branch
   > If the lower bound of this branch is larger than the current upper bound of the UCP, cut this branch;

   > Branch and Bound on this branch;
5. Check the other branch;
6. Report the current solution.

> The End

Review: Computing All PIs

- Goal: simplification
- Why only PIs? (Quine's Theorem)
- How to find all PIs?
  - Quine's tabular method
  - Iterated consensus method

  (if xY and x'Z are PIs, so is their consensus YZ)
  - Recursive method

  (if X is a PI of F = $F_1 \bullet F_2$, then we can rewrite X as Y•Z such that Y and Z are PIs for F_1 and F_2 respectively)

Pseudo-Code: Branch and Bound

1. Apply reduction techniques until they stop;
2. If we find a direct solution with cost less than the upper bound, update upper bound and goto step 9;
3. Calculate the lower/upper bounds;
4. If lower bound is larger than upper bound, goto step 9 and return “no solution”;
5. Pick one column and split into two branches;
6. Branch and Bound on one branch; (recursive call)
7. If the returned solution has cost equal to lower bound, goto step 9;
8. Branch and Bound on the other branch;
9. Report the current solution.

Example of reduction techniques

	wxy	*wxz*	*wyz'*	*wy'z*	*x'y'*	*x'z'*
wx'y'z'					1	1
w'x'y'z					1	
w'x'y'z'					1	1
wxyz	1	1				
wxyz'	1		1			
wx'yz'			1			1
w'x'yz'						1
wxy'z		1		1		
wx'y'z				1	1	

$$\begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 4 \\ 5 \\ 8 \end{array} & \begin{pmatrix} 1 & 1 & & \\ 1 & & 1 & \\ & 1 & & 1 \end{pmatrix} \end{array}$$