Karnaugh Maps (“K-Maps”)

Based on the Theorem:
\[ xA' + xA = x(A' + A) = x \]

This means that two product terms that differ only in one literal (the literal is complemented in one term and uncomplemented in the other) can be grouped to form a single term without that literal. Call terms like this “logically adjacent”.

examples:

\[ w'x'yz + wx'yz = x'y'z \]

\[ ab' + ab = a \]

\[ xyz + xy'z + xy'z' + x'y'z' = xz + y'z' \]
Extending the theorem

The theorem extends to any number of terms that is a power of 2

\[ \text{x'A'B' + xA'B + xAB'} + aAB = xA'(B' + B) + xA(B' + B) \]
\[ = xA' + xA \]
\[ = x \]

Notice that all possible combinations of A and B are included here.

Logically-adjacent minterms

The Karnaugh Map (“K-Map”) is a diagram that arranges the minterms so that if two minterms are Logically Adjacent, they are physically adjacent in the diagram.

For 3 variables:

- m0: 000 \( x'y'z' \)
- m1: 001 \( xy'z \)
- m2: 010 \( xy'z' \)
- m3: 011 \( x'yz \)
- m4: 100 \( x'yz' \)
- m5: 101 \( xy'z \)
- m6: 110 \( xyz' \)
- m7: 111 \( xyz \)

This gives a convenient way to combine logically-adjacent terms.
2-variable Karnaugh Map

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Grouping

We can group 2 adjacent terms on the K-Map. The resulting term contains those literals that are common to all of the grouped terms.

In the example above, we grouped minterms $xy'$ and $xy$.
The result is $x$. 
Grouping

Note that it's OK to use a minterm more than once.

The K-Map gives the same results as algebraic simplification (faster)

Algebraically:
\[ f(x,y) = x'y + xy' + xy \]
\[ = x'y + xy' + xy + xy \]
\[ = x'y + xy + xy' + xy \]
\[ = (x'+x)y + x(y'+y) \]
\[ = y + x \]
\[ f(x,y) = y + x \]
3-variable K-Maps

End cells are adjacent

Example

$$f(x,y,z) = \sum m(2-5)$$

$$f(x,y,z) = xy' + x'y$$
Example 2

\[ f(x, y, z) = \sum m(3, 4, 6, 7) \]

\[ f(x, y, z) = xz' + yz \]

Example 3

\[ f(x, y, z) = \sum m(0, 1, 3, 4, 5) \]

\[ f(x, y, z) = y' + x'z \]
Mapping SOP expressions onto the K-Map

For each product term, write 1 in every cell that is in the "intersection" of the literals in the term.

Note: the bigger the groupings, the simpler the expression.
Note: We must cover all terms, but covering terms more than once increases complexity.

\[ f(x,y,z) = y'z' + xz \]

\[ f(x,y,z) = y'z' + xz + xy' \]

4 variable K-Maps

Top and Bottom Are adjacent

Sides are adjacent

Note that The 4 corners are adjacent to each other
Grouping

\[ f(w, x, y, z) = \sum m(0, 1, 10, 11, 14, 15) \]

\[ f(w, x, y, z) = \overline{w}x'y' + wy \]

Crossing the sides

\[ f(w, x, y, z) = \sum m(0-2, 4-6, 8, 9, 12-14) \]

\[ f(w, x, y, z) = y' + \overline{w}z' + xz' \]
Crossing top and bottom, grouping corners (continued)

\[ F(A,B,C,D) = A'B'C' + B'CD' + A'BCD' + AB'C' \]

\[ F(A,B,C,D) = B'C' + B'D' + A'CD' \]

Grouping definitions

Definitions:

- "Prime Implicant" = "Group"
- "Essential Prime Implicant" = a Prime Implicant that includes at least one minterm that is not included in any other Prime Implicant

B'D' and BD are Essential Prime Implicants

CD and B'C are Prime Implicants, but not Essential Prime Implicants
A grouping process

1. Select all Essential Prime Implicants
2. Select the minimum number of additional Prime Implicants that cover all remaining minterms

\[
f(A,B,C,D) = B'D' + BD + CD
\]

\[
\text{OR-}
\]

\[
f(A,B,C,D) = B'D' + BD + B'C
\]

Same number of literals

Complementing with K-Maps

To simplify the complemented expression, group the 0's

\[
f(w, x, y, z) = \Sigma m(0, 1, 10, 11, 14, 15)
\]

\[
f'(w, x, y, z) = w'y + w'x + wy'
\]
Classroom Examples

Don't cares
“Don’t Cares”

Sometimes we don’t care if a the result of evaluating a Boolean expression for a particular minterm gives a 0 or 1. This often happens because the combination of literals represented by the minterm cannot occur.

Example: \( f(ABCD) = 1 \) if the BCD digit ABCD is even (not 0)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>d</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>d</td>
</tr>
</tbody>
</table>

\( f(ABCD) = \Sigma m(2,4,6,8) + \Sigma d(10-15) \)

Possible to consider don’t cares in grouping:
(1) Treat them as if each d is 1

If we treated all don’t cares as 1:
\( f(ABCD) = \Sigma m(2,4,6,8) + \Sigma d(10-15) \)

\( f(ABCD) = BD' + AB + AD' + CD' \) (8 literals)
Possible ways to consider don’t cares in grouping:

(2) Treat them as if each d is 0

\[ f(ABCD) = \sum m(2,4,6,8) + \sum d(10-15) \]

If we treated all don’t cares as 0:

\[ f(ABCD) = A'CD' + A'BD' + AB'C'D' \] (10 literals)

The best way to consider don’t cares in grouping

\[ f(ABCD) = \sum m(2,4,6,8) + \sum d(10-15) \]

The best approach:
- treat d’s as 1 where they help make larger groups for the “real” 1’s
- treat them as 0 where they do not.

\[ f(ABCD) = CD' + BD' + AD' \] (6 literals)
**Example**

(3.9 in text)

\[ f(\text{wx}yz) = \sum m(1,3,7,11,15) + \sum d(0,2,5) \]

\[ f(\text{wx}yz) = yz + w'x' \]

\[ f(\text{wx}yz) = yz + w'z \]

---

**Classroom Examples**