= input combinations for which $g$ outputs 1

= input combinations for which $f$ outputs 1

**$f$ covers $g$**
if $f=1$ whenever $g=1$

$f \geq g$

= input combinations for which $g$ outputs 1

= input combinations for which $f$ outputs 1

= input combinations for which $h$ outputs 1

If $g$ is a product term & $g \leq f$, Then $g$ is an **implicant** of $f$.

- Removing a literal from any product term (any implicant) makes it cover twice as many minterms.
  - Removing a literal “grows” the term
  - ex. 3 variables: $ab'c$ covers 1 minterm
    $ab'$
    $ac$
    $b'c$
  
  each cover 2 minterms

ab’c is an implicant of $f$.

Any way of removing a literal makes ab’c no longer imply $f$. So ab’c is a **prime implicant** of $f$. 
More Formal Algorithm

- Identify all prime implicants
- Identify all essential ones.
- Circle all essential prime implicants
- Cover the remaining minterms using a minimal number of remaining prime implicants.

Notice that there may be more than one solution. Also notice that the last step is a bit vague 😊

<table>
<thead>
<tr>
<th>ab/c</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

4-variable

<table>
<thead>
<tr>
<th>ab/cd</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

And some practice with these:

<table>
<thead>
<tr>
<th>ab/cd</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ab/cd</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ab/cd</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

What’s left?

- Don’t cares
- 5+ variable
- Product-of-sums
- Programmable techniques
- Lots of practice.
- More context. Remember this is only for 2-level logic...
Don't cares:
Note: I'm leaving the zeros blank to make things more readable!

\[ F = \sum_{w,x,y,z} (1,5,7,11,15) + d(8,12,13,14) \]

Use \( d \) cells to make prime implicants as large as possible.
- No PI should include only \( d \)'s
- Only 1-cells should be considered when finding the minimal cover set.

5-variable

\[ F' = \sum_{A,B,C,D,E} (1,3,15,17,19,29,31) \]

\[ F = \overline{BCE} + BCDE + ABCE \]
Product of Sums:

Let $F$ be:

\[
\begin{array}{c|cccc}
ab/cd & 00 & 01 & 11 & 10 \\
\hline
 00 & 1 & 1 & 1 & 1 \\
 01 & 1 & 1 & 1 & 1 \\
 11 & 1 & 1 & & \\
 10 & 1 & 1 & & \\
\end{array}
\]

Then $F'$ is:

\[
\begin{array}{c|cccc}
ab/cd & 00 & 01 & 11 & 10 \\
\hline
 00 & 1 & & & \\
 01 & & & & \\
 11 & & 1 & 1 & \\
 10 & & 1 & 1 & \\
\end{array}
\]

Find minimal SoP for $F'$.
Use deMorgans.

Programmable techniques

- Later in the semester, time allowing.