Boolean Normal Forms

Truth Table

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>r</th>
<th>( \overline{p} \overline{r} + q + r )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\( \Rightarrow \langle [p, q, r]; [1, 0, 1, 0, 1, 0, 1, 1] \rangle \)

Disjunctive Normal Form (DNF)

\[
\begin{align*}
p q r + \overline{p} q r & \quad + p \overline{q} r + p q r + p q r \\
\Rightarrow & \langle [p, q r]; [000, 010, 100, 110, 111] \rangle \\
\Rightarrow & \langle [p, q r]; [0, 2, 4, 6, 7] \rangle 
\end{align*}
\]

Sum-Of-Products (SOP)

\[
\begin{array}{c|c|c|c|c|c}
 & p q & 00 & 01 & 11 & 10 \\
\hline
0 & 1 & 0 & 1 & 0 & \\
1 & 1 & 0 & 1 & 1 & \\
\end{array}
\]

\( \Rightarrow \overline{p} q + p q + p r \)

Since minimal SOP is not unique, it is not “normal.”
Decision Tree

Decision DAG

Binary Decision Diagram (BDD)
Full-Adder Example

\[ s = \text{majority}(a, b, c) = \overline{abc} + \overline{abc} + \overline{abc} + abc \]
\[ c_o = \text{parity}(a, b, c) = \overline{abc} + \overline{abc} + \overline{abc} + abc \]

\[ s_0 = \overline{abc} + \overline{abc} + \overline{abc} + abc \]
\[ s_1 = \overline{abc} + \overline{abc} + \overline{abc} + abc \]
\[ s_1 = \overline{abc} + \overline{abc} + \overline{abc} + abc \]
\[ s_2 = \overline{abc} + \overline{abc} + \overline{abc} + abc \]

\[ s_2 = c_2 \]
\[ = \overline{abc} + \overline{abc} + \overline{abc} + abc \]
\[ = \overline{abc} + \overline{abc} + \overline{abc} + abc \]
\[ + \ \overline{abc} + \overline{abc} + \overline{abc} + abc \]
\[ + \ \overline{abc} + \overline{abc} + \overline{abc} + abc \]
\[ + \ \overline{abc} + \overline{abc} + \overline{abc} + abc \]
BDD implementation

- BDD\((f)\) is normal, up to variable ordering:

  **Theorem.** \(f \iff g\) iff BDD\((f)\) and BDD\((g)\) are *isomorphic*

- Algorithmic constructions [Bryant 1984] do not use bottom-up reductions.
  - **Cofactoring,** \(f \equiv f_{v=0} + f_{v=1}\).
  - The representation of BDD\((f + g)\) is obtained by a graph merging operation, “or-ing” the representations of BDD\((f)\) and BDD\((g)\).
  - Computer representations are not merely isomorphic, but *identical*. (eq rather than equal) Hashing and caching are used to improve average running times.

- Since boolean equivalence is NP-complete, |BDD\((f)\)| exponential in the worst case.
  - |BDD\((f)\)| is sensitive to the variable ordering.

<table>
<thead>
<tr>
<th>variable ordering</th>
<th>best</th>
<th>worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>n-bit comparison</td>
<td>(n)</td>
<td>(n)</td>
</tr>
<tr>
<td>n-bit addition</td>
<td>(n)</td>
<td>(2^n)</td>
</tr>
<tr>
<td>n-bit multiplication</td>
<td>(2^n)</td>
<td>(2^n)</td>
</tr>
</tbody>
</table>

- Finding a good ordering:
  * NP, of course.
  * Heuristics based on the formula.
  * Incremental, automatic; swapping and other local permutations.
  * External intervention.