2. Representing Boolean Functions

2.1. Representing Boolean Functions.

Definitions 2.1.1.
1. A literal is a Boolean variable or the complement of a Boolean variable.
2. A minterm is a product of literals. More specifically, if there are \( n \) variables, \( x_1, x_2, \ldots, x_n \), a minterm is a product \( y_1 y_2 \cdots y_n \) where \( y_i \) is \( x_i \) or \( \overline{x_i} \).
3. A sum-of-products expansion or disjunctive normal form of a Boolean function is the function written as a sum of minterms.

Discussion

Consider a particular element, say \((0, 0, 1)\), in the Cartesian product \( B^3 \). There is a unique Boolean product that uses each of the variables \( x, y, z \) or its complement (but not both) and has value 1 at \((0, 0, 1)\) and 0 at every other element of \( B^3 \). This product is \( \overline{x} \overline{y} z \).

This expression is called a minterm and the factors, \( \overline{x}, \overline{y}, \) and \( z \), are literals. This observation makes it clear that one can represent any Boolean function as a sum-of-products by taking Boolean sums of all minterms corresponding to the elements of \( B^n \) that are assigned the value 1 by the function. This sum-of-products expansion is analogous to the disjunctive normal form of a propositional expressions discussed in Propositional Equivalences in MAD 2104.

2.2. Example 2.2.1.

Example 2.2.1. Find the disjunctive normal form for the Boolean function \( F \) defined by the table

<table>
<thead>
<tr>
<th>( x ), ( y ), ( z )</th>
<th>( F(x, y, z) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 0, 0</td>
<td>0</td>
</tr>
<tr>
<td>0, 0, 1</td>
<td>0</td>
</tr>
<tr>
<td>0, 1, 0</td>
<td>1</td>
</tr>
<tr>
<td>0, 1, 1</td>
<td>0</td>
</tr>
<tr>
<td>1, 0, 0</td>
<td>1</td>
</tr>
<tr>
<td>1, 0, 1</td>
<td>1</td>
</tr>
<tr>
<td>1, 1, 0</td>
<td>0</td>
</tr>
<tr>
<td>1, 1, 1</td>
<td>0</td>
</tr>
</tbody>
</table>

Solution: \( F(x, y, z) = xy\overline{z} + x\overline{y}z + x\overline{y}z \)
Discussion

The disjunctive normal form should have three minterms corresponding to the three triples for which \( F \) takes the value 1. Consider one of these: \( F(0,1,0) = 1 \). In order to have a product of literals that will equal 1, we need to multiply literals that have a value of 1. At the triple \((0,1,0)\) the literals we need are \( x, y, \) and \( z \), since \( x = y = z = 1 \) when \( x = 0, y = 1, \) and \( z = 0 \). The corresponding minterm, \( xy\bar{z} \), will then have value 1 at \((0,1,0)\) and 0 at every other triple in \( B^3 \). The other two minterms come from considering \( F(1,0,0) = 1 \) and \( F(1,0,1) = 1 \). The sum of these three minterms will have value 1 at each of \((1,0,0), (0,1,0), (1,0,1)\) and 0 at all other triples in \( B^3 \).

\[2.3. \text{Example 2.3.1.}\]

\textbf{Example 2.3.1.} Simply the expression
\[ F(x, y, z) = xy \bar{z} + x \bar{y} \bar{z} + x \bar{y}z \]

using properties of Boolean expressions.

\textbf{Solution.}
\[
xy \bar{z} + x \bar{y} \bar{z} + x \bar{y}z = xy \bar{z} + x \bar{y} (\bar{z} + z) \\
= xy \bar{z} + x \bar{y} \cdot 1 \\
= xy \bar{z} + x \bar{y}
\]

Discussion

Example 2.3.1 shows how we might simplify the function we found in Example 2.2.1. Often sum-of-product expressions may be simplified, but any nontrivial simplification will produce an expression that is not in sum-of-product form. A sum-of-products form must be a sum of minterms and a minterm must have each variable or its compliment as a factor.

\textbf{Example 2.3.2.} The following are examples of “simplifying” that changes a sum-of-products to an expression that is not a sum-of-products:

- \[ \text{sum-of-product form: } x \bar{y}z + x \bar{y} \bar{z} + xyz \]
- \[ \text{NOT sum-of-product form: } = x \bar{y} + xyz \]
- \[ \text{NOT sum-of-product form: } = x(\bar{y} + yz) \]

\textbf{Exercise 2.3.1.} Find the disjunctive normal form for the Boolean function, \( G \), of degree 4 such that \( G(x_1, x_2, x_3, x_4) = 0 \) if and only if at least 3 of the variables are 1.
2.4. Functionally Complete.

Definition 2.4.1. A set of operations is called **functionally complete** if every Boolean function can be expressed using only the operations in the set.

Discussion

Since every Boolean function can be expressed using the operations \( \{+, \cdot, \overline{\cdot}\} \), the set \( \{+, \cdot, \overline{\cdot}\} \) is **functionally complete**. The fact that every function may be written as a sum-of-products demonstrates that this set is functionally complete.

There are many other sets that are also functionally complete. If we can show each of the operations in \( \{+, \cdot, \overline{\cdot}\} \) can be written in terms of the operations in another set, \( S \), then the set \( S \) is functionally complete.

2.5. Example 2.5.1.

**Example 2.5.1.** Show that the set of operations \( \{\cdot, \overline{\cdot}\} \) is functionally complete.

**Proof.** Since \( \cdot \) and \( \overline{\cdot} \) are already members of the set, we only need to show that + may be written in terms of \( \cdot \) and \( \overline{\cdot} \).

We claim

\[ x + y = \overline{x} \cdot \overline{y}. \]

Proof of Claim Version 1

\begin{align*}
\overline{x} \cdot \overline{y} &= \overline{x + y} \quad \text{De Morgan’s Law} \\
&= x + y \quad \text{Law of Double Complement}
\end{align*}

Proof of Claim Version 2

\[
\begin{array}{|c|c|c|c|c|c|c|c|}
\hline
x & y & \overline{x} & \overline{y} & x \cdot y & \overline{x} \cdot \overline{y} & x + y \\
\hline
1 & 1 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 \\
\hline
\end{array}
\]

Discussion

**Exercise 2.5.1.** Show that \( \{+, \overline{\cdot}\} \) is functionally complete.
Exercise 2.5.2. Prove that the set \{+, \cdot\} is not functionally complete by showing that the function \(F(x) = \overline{x}\) (of order 1) cannot be written using only \(x\) and addition and multiplication.

2.6. NAND and NOR.

Definitions 2.6.1.
1. The binary operation NAND, denoted \(|\), is defined by the table

| \(x\) | \(y\) | \(x|y\) |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 1 |

2. The binary operation NOR, denoted \(\downarrow\), is defined by the table

<table>
<thead>
<tr>
<th>(x)</th>
<th>(y)</th>
<th>(x \downarrow y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Discussion

Notice the NAND operator may be thought of as “not and” while the NOR may be thought of as “not or.”

Exercise 2.6.1. Show that \(x|y = \overline{x \cdot y}\) for all \(x\) and \(y\) in \(B = \{0, 1\}\).

Exercise 2.6.2. Show that \{|\} is functionally complete.

Exercise 2.6.3. Show that \(x \downarrow y = \overline{x + y}\) for all \(x\) and \(y\) in \(B = \{0, 1\}\).

Exercise 2.6.4. Show that \{|\} is functionally complete.