By defining logic gates based on Boolean algebra, we can use algebraic methods to manipulate circuits.

- **Sum**: $a + b$
- **Product**: $a \cdot b$
- **Literal**: $a$, $b$, $c$
- **Variable**: $a$, $b$, $c$
- **Product term**: $a \cdot b \cdot c$
- **Sum-of-products**: $a + b + c$
- **Complement**: $a'$
- **Example equation**: $F(a,b,c) = a'bc + abc' + ab + c$

**Boolean Algebra Terminology**

- **Example equation**: $F(a,b,c) = a'bc + abc' + ab + c$
- **Variable**:
  - Represents a value (0 or 1)
  - Three variables: $a$, $b$, and $c$
- **Literal**:
  - Appearance of a variable, in true or complemented form
- **Product term**:
  - Product of literals
  - Four product terms: $a \cdot b, a' \cdot c, a \cdot b, c$
- **Sum-of-products**:
  - Equation written as OR of product terms only
  - Above equation is in sum-of-products form: $F = (a+b)c + d'$ is not.

**Boolean Algebra Properties**

- **Commutative**:
  - $a \cdot b = b \cdot a$
- **Distributive**:
  - $a \cdot (b + c) = a \cdot b + a \cdot c$
  - $a + (b \cdot c) = (a + b) \cdot (a + c)$
- **Associative**:
  - $(a + b) + c = a + (b + c)$
  - $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
- **Identity**:
  - $0 \cdot a = 0, 1 + a = a$
- **Complement**:
  - $a + a' = 1, a \cdot a' = 0$
- **To prove, just evaluate all possibilities**

**Example uses of the properties**

- **Show $abc$ equivalent to $bca$**:
  - Use commutative property:
    - $a \cdot b \cdot c = b \cdot c \cdot a = c \cdot b \cdot a$
- **Show $abc = abc$**:
  - Use first distributive property:
    - $a \cdot b \cdot c = (a \cdot b) \cdot c$
  - **Complement property**:
    - Replace $a'$ by $1$, $abc(a') = ab1$
  - **Identity property**:
    - $(a' \cdot 1) = a$
  - **Show $x \cdot z$ equivalent to $x + z$**:
    - Second distributive property:
      - $x \cdot z = (x + y) \cdot (z + y)$
    - **Complement property**:
      - Replace $x'y$ by $1$, $x'y(y' + z')$
    - **Identity property**:
      - Replace $(y + z)$ by $y + z$.
Example that Applies Boolean Algebra Properties

- Want automatic door opener circuit (e.g., for grocery store)
  - Output: \( f=1 \) opens door
  - Inputs:
    - \( p=1 \): person detected
    - \( h=1 \): switch forcing hold open
    - \( c=1 \): key forcing closed
  - Want open door when
    - \( h=1 \) and \( c=0 \), or
    - \( h=0 \) and \( p=1 \) and \( c=0 \)
  - Equation: \( f = hc' + h'pc' \)

Can the circuit be simplified?

\[
\begin{align*}
\text{DoorOpener} & \quad \text{Simplified circuit} \\
\begin{array}{c}
\text{f} \\
\text{h} \\
\text{c} \\
\text{p}
\end{array}
\end{align*}
\]

Example that Applies Boolean Algebra Properties

Video

Boolean Algebra: Additional Properties

- Null elements
  - \( a + 1 = 1 \)
  - \( a * 0 = 0 \)
- Idempotent Law
  - \( a + a = a \)
  - \( a * a = a \)
- Involution Law
  - \( (a')' = a \)
- DeMorgan's Law
  - \( (a + b)' = a'b' \)
  - \( (ab)' = a' + b' \)
  - Very useful!
- To prove, just evaluate all possibilities

Example Applying DeMorgan's Law

Aircraft lavatory sign example

- Behavior
  - Three lavatories, each with sensor (a, b, c), equals 1 if door locked
  - Light "Available" sign (S) if any lavatory available
- Equation and circuit
  - \( S = a' + b' + c' \)
- Transform
  - \( (abc)' = a' + b' + c' \) (by DeMorgan's Law)
  - \( S = (abc)' \)
- New circuit

Example Applying Properties

- For door opener \( f = c'(h+p) \), prove door stays closed \( (f=0) \) when \( c=1 \)
  - \( f = c'(h+p) \)
  - Let \( c = 1 \) (door forced closed)
  - \( f = 1'(h+p) \)
  - \( f = 0(h+p) \)
  - \( f = 0h + 0p \) (by the distributive property)
  - \( f = 0 \) (by the null elements property)

Commutative
- \( a + b = b + a \)
- \( a * b = b * a \)

Distributive
- \( a + (b + c) = (a + b) + (a + c) \)
- \( a * (b + c) = (a * b) + (a * c) \)

Associative
- \( a + b + c = (a + b) + c = a + (b + c) \)
- \( a * b * c = (a * b) * c = a * (b * c) \)

Null elements
- \( a + 0 = a \)
- \( a * 1 = a \)

Complement
- \( a + a' = 1 \)
- \( a * a' = 0 \)

Idempotent Law
- \( a + a = a \)
- \( a * a = a \)

Involution Law
- \( (a')' = a \)

DeMorgan's Law
- \( (a + b)' = a'b' \)
- \( (ab)' = a' + b' \)

Null elements
- \( a + 0 = a \)
- \( a * 1 = a \)

Complement
- \( a + a' = 1 \)
- \( a * a' = 0 \)

Identity
- \( a + 0 = a \)
- \( a * 1 = a \)

Involution Law
- \( (a')' = a \)

DeMorgan's Law
- \( (a + b)' = a'b' \)
- \( (ab)' = a' + b' \)
Complement of a Function

- Commonly want to find complement (inverse) of function $F$:
  - 0 when $F$ is 1; 1 when $F$ is 0.
- Use DeMorgan’s Law repeatedly:
  - Note: DeMorgan’s Law defined for more than two variables, e.g.:
    - $(a + b + c)' = a'b'c'$
    - $(abc)' = (a' + b' + c')$
- Complement of $f = w'xy + wx'y'z'$:
  - $f' = (w'xy + wx'y'z')'$
  - $f' = (w'xy)'(wx'y')'$ (by DeMorgan’s Law)
  - $f' = (w + x + y')(w' + x + y + z)$ (by DeMorgan’s Law)
- Can then expand into sum-of-products form

Example that Applies Boolean Algebra Properties

Video

Representations of Boolean Functions

- A function can be represented in different ways
  - Above shows seven representations of the same functions $F(a,b)$, using four different methods: English, Equation, Circuit, and Truth Table

Converting among Representations

- Can convert from any representation to another
- Common conversions:
  - Equation to circuit (we did this earlier)
  - Circuit to equation:
    - Start at inputs, write expression of each gate output

Truth Table Representation of Boolean Functions

- Define value of $F$ for each possible combination of input values:
  - 2-input function: 4 rows
  - 3-input function: 8 rows
  - 4-input function: 16 rows
- Q: Use truth table to define function $F(a,b,c)$ that is 1 when $abc$ is 5 or greater in binary
Example: Converting from Truth Table to Equation

- Parity bit: Extra bit added to data, intended to enable detection of error (a bit changed unintentionally)
  - e.g., errors can occur on wires due to electrical interference
- Even parity: Set parity bit so total number of 1s (data + parity) is even
  - e.g., if data is 001, parity bit is 1 → 0011 has even number of 1s
- Want equation, but easiest to start from truth table for this example

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Convert to eqn.

\[ P = a'b'c + a'bc' + ab'c' + abc \]

Example: Converting from Circuit to Truth Table

- First convert to circuit to equation, then equation to table

<table>
<thead>
<tr>
<th>F</th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Standard Representation: Truth Table

- How can we determine if two functions are the same?
  - Recall automatic door example
- Same as \( f = hc' + h'pc' \)?
- Used algebraic methods
- But if we failed, does that prove not equal? No.
- Solution: Convert to truth tables
  - Only ONE truth table representation of a given function
- Standard representation—for given function, only one version in standard form exists

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Q. Determine if \( F = ab + a' \) is same function as \( F = a'b' + a'b + ab \), by converting each to truth table first

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Canonical Form

- Truth tables too big for numerous inputs
- Use standard form of equation instead
  - Known as canonical form
- Regular algebra: group terms of polynomial by power
  - \( ax^2 + bx + c \) (3x^2 + 4x + 2x^2 + 3 + 1 \rightarrow 5x^2 + 4x + 4)
- Boolean algebra: create sum of minterms
  - Minterms: product term with every function literal appearing exactly once, in true or complemented form
  - Just multiply-out equation until sum of product terms
  - Then expand each term until all terms are minterms

Q. Determine if \( F = (a \land b) + a' \) is equivalent to \( F = (a \land b) + a' + b + ab \), by converting first equation to canonical form (second already is)

\[ F = ab + a' \] (already sum of products)
\[ F = ab + a' + b' \] (expanding term)
\[ F = ab + a' + b \] (Equivalent – same three terms as other equation)
Canonical Form – Sum of Minterms

- Q: Determine whether the functions \( G(a,b,c,d,e) = abc + a'bcde \) and 
  \( H(a,b,c,d,e) = abde + abde' + a'bcde + a'bcde(a' + c) \) are equivalent.

\[
G = abcd + a'bcde \\
G = abcd(e+e') + a'bcde \\
G = abde + abde' + a'bcde \quad (\text{sum of minterms form})
\]

\[
H = abcde + abcde' + a'bcde + a'bcde(a' + c) \\
H = abcde + abcde' + a'bcde + a'bcdea' + a'bcdec \\
H = abcde + abcde' + a'bcde + a'bcde + a'bcde \\
H = a'bcde + abcde' + abcde
\]

Equivalent

Compact Sum of Minterms Representation

- List each minterm as a number
- Number determined from the binary representation of its variables’ values
  - \( a'bcde \) corresponds to 01111, or 15
  - \( abcde' \) corresponds to 11110, or 30
  - \( abcde \) corresponds to 11111, or 31
- Thus, \( H = a'bcde + abcde' + abcde \) can be written as:
  - \( H = \sum m(15,30,31) \)
  - "H is the sum of minterms 15, 30, and 31"

Multiple-Output Circuits

- Many circuits have more than one output
- Can give each a separate circuit, or can share gates
- Ex: \( F = ab + c' \), \( G = ab + bc \)

Option 1: Separate circuits

Option 2: Shared gates

Multiple-Output Example: BCD to 7-Segment Converter

Example: Multiple-Output Circuits
Summary

• Boolean Operator
  - (), ', *, +.
  - precedence.

• Terminology
  - variable, literal, product term, sum-of-products.

• Properties
  - Commutative, distributive, associative, identity, complement.
  - Null elements, idempotent, involution, DeMorgan.

• Representation
  - Truth table, Boolean equation, circuit.
  - Canonical form, multiple-output circuits.