Outline

1 Switching Circuits

2 Boolean Algebra
   • Examples

3 Algebraic Equivalence
   • Examples

4 Sets connection with Boolean Algebra
Let’s Start!

- Switching circuits are a way of describing pictorially the symbolic logic that you met earlier.
Let’s Start!

- Switching circuits are a way of describing pictorially the symbolic logic that you met earlier.
- Boolean algebras are abstract mathematical constructions that unify the apparently different concepts of sets, symbolic logic and switching systems.
A switch is a device which is attached to a point in an electrical circuit.
A switch is a device which is attached to a point in an electrical circuit. The switch can be in either of two states, open or closed:
Switches

- A **switch** is a device which is attached to a point in an electrical circuit.
- The switch can be in either of two states, **open** or **closed**:
  - in the **open** state the switch does not allow current to flow through the point.
A **switch** is a device which is attached to a point in an electrical circuit. The switch can be in either of two states, **open** or **closed**:

- in the **open** state the switch does not allow current to flow through the point.
- in the **closed** state the switch does allow current to flow through the point.
Switches

- A **switch** is a device which is attached to a point in an electrical circuit.
- The switch can be in either of two states, **open** or **closed**:
  - in the **open** state the switch does not allow current to flow through the point.
  - in the **closed** state the switch does allow current to flow through the point.
- We shall indicate a switch by means of the symbols

```
open closed
x x
```

In principle, "x" indicates a sentence such that the associated switch is closed when x is true and it is open when x is false. Another notation is –lx–.
A **switch** is a device which is attached to a point in an electrical circuit.

The switch can be in either of two states, **open** or **closed**:

- in the **open** state the switch does not allow current to flow through the point.
- in the **closed** state the switch does allow current to flow through the point.

We shall indicate a switch by means of the symbols

![Switch Symbols]

*In principle, "x" indicates a sentence such that the associated switch is closed when \( x \) is true and it is open when \( x \) is false. Another notation is \( \neg(x) \).*
Switching Circuits

Two points (available to the outside) are connected by a switching circuit if and only if they are connected by wires on which a finite collection of switches are located.
Switching Circuits

- Two points (available to the outside) are connected by a **switching circuit** if and only if they are connected by wires on which a finite collection of switches are located.
- For example, the following

![Diagram showing a switching circuit involving points x, y, and z, a battery, and a light bulb, connected by switches.]

...is a **switching circuit**, making use of an energy source (battery) an output (light) as well as a switching system.
If switches $x$ and $z$ are open while switch $y$ is closed, then the state of the switching system may be represented by

\[ x \quad y \quad z \]

In order to describe switching systems formally and mathematically, we denote open and closed states by 0 and 1, respectively. It's obvious that the state space $S$ for any switch or switching system is composed of two states: 0 (open) and 1 (closed), i.e., $S = \{0, 1\}$. 

![Diagram of switching circuits]
If switches $x$ and $z$ are open while switch $y$ is closed, then the state of the switching system may be represented by $x\ y\ z$.

In order to describe switching systems formally and mathematically, we denote open and closed states by 0 and 1, respectively.
Switching Circuits

- If switches $x$ and $z$ are open while switch $y$ is closed, then the state of the switching system may be represented by $x \ y \ z$

- In order to describe switching systems formally and mathematically, we denote open and closed states by 0 and 1, respectively.
- It’s obvious that the state space $S$ for any switch or switching system is composed of two states: 0 (open) and 1 (closed), i.e., $S = \{0, 1\}$. 
Switching Circuits

In a switching system, switches may be connected with one another

▶ in parallel: current flows between points $a$ and $b$ iff $x \lor y$ is true

$\quad x \quad y$

▶ in series: current flows between points $a$ and $b$ iff $x \land y$ is true

$\quad x \quad y$

▶ through the use of complementary switches: for any given switch $x$, the corresponding complementary switch, denoted by $x'$, is always in the opposite state to that of $x$.

$\quad x$

$\quad x'$
Switching Circuits

- Easy generalisation to the case of a finite number of switches $x_1, x_2, \ldots, x_n$:
  - Connected in parallel: current flows through the circuit iff $x_1 \lor x_2 \lor \ldots \lor x_n$ is true.
  - Connected in series: current flows through the circuit iff $x_1 \land x_2 \land \ldots \land x_n$ is true.

For any two switches $x$ and $y$, we use $x + y$ and $x \cdot y$ to denote their parallel and series connections respectively. The symbols "." and "+" here shouldn’t be confused with those in the arithmetic, although there exist some similarities.
Switching Circuits

- Easy generalisation to the case of a finite number of switches $x_1, x_2, \ldots, x_n$:
  - connected in parallel:
    current flows through the circuit iff $x_1 \lor x_2 \lor \ldots \lor x_n$ is true.
Switching Circuits

- Easy generalisation to the case of a finite number of switches $x_1, x_2, \ldots, x_n$:
  - connected in parallel:
    current flows through the circuit iff $x_1 \lor x_2 \lor \ldots \lor x_n$ is true.
  - connected in series:
    current flows through the circuit iff $x_1 \land x_2 \land \ldots \land x_n$ is true.

For any two switches $x$ and $y$, we use $x + y$ and $x \cdot y$ to denote their parallel and series connections respectively. The symbols "." and "+" here shouldn't be confused with those in the arithmetic, although there exist some similarities.
Switching Circuits

- Easy generalisation to the case of a finite number of switches $x_1, x_2, \ldots, x_n$:
  - connected in parallel:
    current flows through the circuit iff $x_1 \lor x_2 \lor \ldots \lor x_n$ is true.
  - connected in series:
    current flows through the circuit iff $x_1 \land x_2 \land \ldots \land x_n$ is true.

- For any two switches $x$ and $y$, we use $x + y$ and $x \cdot y$ to denote their parallel and series connections respectively.
Switching Circuits

- Easy generalisation to the case of a finite number of switches $x_1, x_2, \ldots, x_n$:
  - connected in parallel: current flows through the circuit iff $x_1 \lor x_2 \lor \ldots \lor x_n$ is true.
  - connected in series: current flows through the circuit iff $x_1 \land x_2 \land \ldots \land x_n$ is true.

- For any two switches $x$ and $y$, we use $x + y$ and $x \cdot y$ to denote their parallel and series connections respectively.

- The symbols ”.” and ”+” here shouldn’t be confused with those in the arithmetic, although there exist some similarities.
The state of compound switches

- With the above introduced notations, we can represent the state of, or the effect of switches in parallel, in series and so on by the following tables.

<table>
<thead>
<tr>
<th>Parallel</th>
<th>Series</th>
<th>Complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$y$</td>
<td>$x + y$</td>
</tr>
<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>

- Switch $y$ is open.
- Switch $x$ is closed.
- Switching system with $x$ and $y$ in parallel is closed.
- As the special case in the above tables we have in particular $1 + 1 = 1$.
- Notice a similarity between these tables and truth tables: here 0 can be considered as a F(alse) and 1 as a T( rue).
- The first two columns are constructed in a similar way to truth tables (with F's first instead of T's) to capture all possible combinations of 0's and 1's.
The state of compound switches

- With the above introduced notations, we can represent the state of, or the effect of switches in parallel, in series and so on by the following tables.

<table>
<thead>
<tr>
<th>Parallel</th>
<th>Series</th>
<th>Complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$ $y$</td>
<td>$x + y$</td>
<td>$x$ $y$</td>
</tr>
<tr>
<td>0 0</td>
<td>0</td>
<td>0 0</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
<td>0 1</td>
</tr>
<tr>
<td>1 0</td>
<td>1</td>
<td>1 0</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
<td>1 1</td>
</tr>
</tbody>
</table>

- switch $x$ is closed
- switch $y$ is open
- switching system with $x$ and $y$ in parallel is closed
- As the special case in the above tables we have in particular $1 + 1 = 1$. 

Notice a similarity between these tables and truth tables: here 0 can be considered as a F(alse) and 1 as a T( rue).

The first two columns are constructed in a similar way to truth tables (with F's first instead of T's) to capture all possible combinations of 0's and 1's.
The state of compound switches

- With the above introduced notations, we can represent the state of, or the effect of switches in parallel, in series and so on by the following tables

<table>
<thead>
<tr>
<th>Parallel</th>
<th>Series</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$x + y$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Notice a similarity between these tables and truth tables: here 0 can be considered as a F(alse) and 1 as a T(rue).

- As the special case in the above tables we have in particular $1 + 1 = 1$.

- Switching system with $x$ and $y$ in parallel is closed:
  - switch $x$ is closed
  - switch $y$ is open

- Switching system with $x$ and $y$ in series is closed:
  - switching system with $x$ and $y$ in parallel is closed

- Complement:

<table>
<thead>
<tr>
<th>Complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
The state of compound switches

- With the above introduced notations, we can represent the state of, or the effect of switches in parallel, in series and so on by the following tables:

<table>
<thead>
<tr>
<th>Parallel</th>
<th>Series</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$y$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- The first two columns are constructed in a similar way to truth tables (with F’s first instead of T’s) to capture all possible combinations of 0’s and 1’s.
- As the special case in the above tables we have in particular $1 + 1 = 1$.
- Notice a similarity between these tables and truth tables: here 0 can be considered as a F(alse) and 1 as a T( rue).

switch $x$ is closed

switch $y$ is open

switching system with $x$ and $y$ in parallel is closed

Complement

<table>
<thead>
<tr>
<th>$x$</th>
<th>$x'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
A **Boolean algebra** is essentially a mathematical abstraction and extension of switching systems and/or symbolic logic and the foundation of electronic design of computers.
A **Boolean algebra** is essentially a mathematical abstraction and extension of switching systems and/or symbolic logic and the foundation of electronic design of computers.

It is a branch of mathematics developed around 1850 by George Boole as rules of *algebra* for logical thinking.
Boolean Algebra

- A Boolean algebra is essentially a mathematical abstraction and extension of switching systems and/or symbolic logic and the foundation of electronic design of computers.
- It is a branch of mathematics developed around 1850 by George Boole as rules of algebra for logical thinking.
- Two preliminary definitions:
  - A binary operation on a set $A$ is a mapping $f: A \times A \rightarrow A$, i.e., $f(a, b) \in A$ for any pair $(a, b)$ with $a$ and $b$ both in $A$.
  - So a binary operation takes two elements of a set and produces a third.
  - A unary operation on a set $A$ is a mapping $f: A \rightarrow A$, i.e., $f(a) \in A$ for any $a$ in $A$.
  - So a unary operation takes one element of a set and produces another.

Example
For switching systems with state space $S = \{0, 1\}$, the $+$ and $\cdot$ operation are binary and the $'$ operation is unary.

Solution.
This is because for any switching systems $x$ and $y$, we have that $x + y$, $x \cdot y$ and $x'$ are all still switching systems with the same state space $S$. 

Note.
Binary operator or operation has nothing to do with binary numbers.
Boolean Algebra

- A **Boolean algebra** is essentially a mathematical abstraction and
  extension of switching systems and/or symbolic logic and the foundation
  of electronic design of computers.
- It is a branch of mathematics developed around 1850 by George Boole as
  rules of *algebra* for logical thinking.
- Two preliminary definitions:
  - A **binary operation** on a set $A$ is a mapping $f: A \times A \to A$, i.e.,
    $f(a, b) \in A$ for any pair $(a, b)$ with $a$ and $b$ both in $A$.
  - A **unary operation** on a set $A$ is a mapping $f: A \to A$, i.e.,
    $f(a) \in A$ for any $a$ in $A$.

Example

For switching systems with state space $S = \{0, 1\}$, the $+$ and $\cdot$ operation
are binary and the $'$ operation is unary.

Solution.

This is because for any switching systems $x$ and $y$, we have that $x + y,$ $x \cdot y$
and $x'$ are all still switching systems with the same state space $S$.

Note.

Binary operator or operation has nothing to do with binary numbers.
Boolean Algebra

A Boolean algebra is essentially a mathematical abstraction and extension of switching systems and/or symbolic logic and the foundation of electronic design of computers.

It is a branch of mathematics developed around 1850 by George Boole as rules of algebra for logical thinking.

Two preliminary definitions:

- A binary operation on a set $A$ is a mapping $f: A \times A \rightarrow A$, i.e., $f(a, b) \in A$ for any pair $(a, b)$ with $a$ and $b$ both in $A$.
  - So a binary operation takes two elements of a set and produces a third.

- A unary operation on a set $A$ is a mapping $f: A \rightarrow A$, i.e., $f(a) \in A$ for any $a$ in $A$.

Example

For switching systems with state space $S = \{0, 1\}$, the $+$ and $\cdot$ operation are binary and the $'$ operation is unary.

Solution.

This is because for any switching systems $x$ and $y$, we have that $x + y$, $x \cdot y$ and $x'$ are all still switching systems with the same state space $S$.

Note.

Binary operator or operation has nothing to do with binary numbers.
**Boolean Algebra**

- A **Boolean algebra** is essentially a mathematical abstraction and extension of switching systems and/or symbolic logic and the foundation of electronic design of computers.
- It is a branch of mathematics developed around 1850 by George Boole as rules of *algebra* for logical thinking.
- Two preliminary definitions:
  - A **binary operation** on a set $A$ is a mapping $f: A \times A \to A$, i.e., $f(a, b) \in A$ for any pair $(a, b)$ with $a$ and $b$ both in $A$.
    - So a binary operation takes two elements of a set and produces a third.
  - A **unary operation** on a set $A$ is a mapping $f: A \to A$, i.e., $f(a) \in A$ for any $a$ in $A$.
Boolean Algebra

- A Boolean algebra is essentially a mathematical abstraction and extension of switching systems and/or symbolic logic and the foundation of electronic design of computers.
- It is a branch of mathematics developed around 1850 by George Boole as rules of algebra for logical thinking.
- Two preliminary definitions:
  - A **binary operation** on a set $A$ is a mapping $f: A \times A \rightarrow A$, i.e., $f(a, b) \in A$ for any pair $(a, b)$ with $a$ and $b$ both in $A$.
    - So a binary operation takes two elements of a set and produces a third.
  - A **unary operation** on a set $A$ is a mapping $f: A \rightarrow A$, i.e., $f(a) \in A$ for any $a$ in $A$.
    - So a unary operation takes one element of a set and produces another.
Boolean Algebra

- A Boolean algebra is essentially a mathematical abstraction and extension of switching systems and/or symbolic logic and the foundation of electronic design of computers.
- It is a branch of mathematics developed around 1850 by George Boole as rules of algebra for logical thinking.
- Two preliminary definitions:
  - A **binary operation** on a set $A$ is a mapping $f: A \times A \rightarrow A$, i.e., $f(a, b) \in A$ for any pair $(a, b)$ with $a$ and $b$ both in $A$.
    - So a binary operation takes two elements of a set and produces a third.
  - A **unary operation** on a set $A$ is a mapping $f: A \rightarrow A$, i.e., $f(a) \in A$ for any $a$ in $A$.
    - So a unary operation takes one element of a set and produces another.

Example

For switching systems with state space $S = \{0, 1\}$, the $+$ and $\cdot$ operation are binary and the $'$ operation is unary.

Solution.

This is because for any switching systems $x$ and $y$, we have that $x + y$, $x \cdot y$ and $x'$ are all still switching systems with the same state space $S$. 

Note.

Binary operator or operation has nothing to do with binary numbers.
Boolean Algebra

- A **Boolean algebra** is essentially a mathematical abstraction and extension of switching systems and/or symbolic logic and the foundation of electronic design of computers.
- It is a branch of mathematics developed around 1850 by George Boole as rules of *algebra* for logical thinking.
- Two preliminary definitions:
  - A **binary operation** on a set $A$ is a mapping $f: A \times A \rightarrow A$, i.e., $f(a, b) \in A$ for any pair $(a, b)$ with $a$ and $b$ both in $A$.
    - So a binary operation takes two elements of a set and produces a third.
  - A **unary operation** on a set $A$ is a mapping $f: A \rightarrow A$, i.e., $f(a) \in A$ for any $a$ in $A$.
    - So a unary operation takes one element of a set and produces another.

**Example**

For switching systems with state space $S=\{0,1\}$, the “+” and “·” operation are binary and the “′” operation is unary.
Boolean Algebra

- **A Boolean algebra** is essentially a mathematical abstraction and extension of switching systems and/or symbolic logic and the foundation of electronic design of computers.
- It is a branch of mathematics developed around 1850 by George Boole as rules of *algebra* for logical thinking.
- Two preliminary definitions:
  - A **binary operation** on a set $A$ is a mapping $f: A \times A \rightarrow A$, i.e., $f(a, b) \in A$ for any pair $(a, b)$ with $a$ and $b$ both in $A$.
    - So a binary operation takes two elements of a set and produces a third.
  - A **unary operation** on a set $A$ is a mapping $f: A \rightarrow A$, i.e., $f(a) \in A$ for any $a$ in $A$.
    - So a unary operation takes one element of a set and produces another.

**Example**

For switching systems with state space $S=\{0, 1\}$, the “+” and “·” operation are binary and the “′” operation is unary.

**Solution.** This is because for any switching systems $x$ and $y$, we have that $x + y$, $x \cdot y$ and $x'$ are all still switching systems with the same state space $S$.

**Note.** Binary operator or operation has nothing to do with binary numbers.
Boolean Algebra

Definition

A **Boolean algebra** is a set \( S \) on which are defined two binary operations \( + \) and \( \cdot \) and one unary operation \( ' \) and in which there are at least two distinct elements 0 and 1 such that the following properties hold for all \( a, b, c \in S \):

B1. \[ a + b = b + a \] \[ a \cdot b = b \cdot a \] \} **commutativity**
B2. \[ (a + b) + c = a + (b + c) \] \[ (a \cdot b) \cdot c = a \cdot (b \cdot c) \] \} **associativity**
B3. \[ a + (b \cdot c) = (a + b) \cdot (a + c) \] \[ a \cdot (b + c) = (a \cdot b) + (a \cdot c) \] \} **distributivity**
B4. \[ a + 0 = a \] \[ a \cdot 1 = a \] \} **identity relations**
B5. \[ a + a' = 1 \] \[ a \cdot a' = 0 \] \} **complementation**
Boolean Algebra

- The "0" and "1" in the above are just a notation, two special elements of $S$.

▶ For example, $a + b' \cdot c$ in fact means $a + ((b' \cdot c))$.

Each property in the definition of a Boolean algebra has its dual as part of the definition.
▶ The dual is obtained by interchanging $+$ with $\cdot$ and 0 with 1.
▶ Unary operation $'$ remains unchanged.
Boolean Algebra

- The "0" and "1" in the above are just a notation, two special elements of $S$.
- They are symbols rather than normal numerical values.
Boolean Algebra

- The ”0” and ”1” in the above are just a notation, two special elements of $S$.
- They are symbols rather than normal numerical values.
- Likewise the operators “+”, “·” and “′” are also symbols, each representing the designated special roles.
Boolean Algebra

- The "0" and "1" in the above are just a notation, two special elements of $S$.
- They are *symbols* rather than normal numerical values.
- Likewise the operators "+", "·" and "′" are also symbols, each representing the designated special roles.
- A Boolean algebra is thus often represented by a *tuple* $(S, +, ·, ′, 0, 1)$ which carries all the relevant components.
Boolean Algebra

- The ”0” and ”1” in the above are just a notation, two special elements of $S$.
- They are *symbols* rather than normal numerical values.
- Likewise the operators “+”, “·” and “′” are also symbols, each representing the *designated* special roles.
- A Boolean algebra is thus often represented by a *tuple* $(S, +, ·, ′, 0, 1)$ which carries all the relevant components.
- A variable $x \in S$ is called a **Boolean variable**.
Boolean Algebra

- The ”0” and ”1” in the above are just a notation, two special elements of $S$.
- They are symbols rather than normal numerical values.
- Likewise the operators “+”, “.” and “′” are also symbols, each representing the designated special roles.
- A Boolean algebra is thus often represented by a tuple $(S, +, \cdot, ′, 0, 1)$ which carries all the relevant components.
- A variable $x \in S$ is called a Boolean variable.
- A combination of some elements of $S$ (variables) via the connectives $+$ and $\cdot$ and the complement $′$ is a Boolean expression.
Boolean Algebra

- The ”0” and ”1” in the above are just a notation, two special elements of $S$.
- They are symbols rather than normal numerical values.
- Likewise the operators “+”, “·” and “′” are also symbols, each representing the designated special roles.
- A Boolean algebra is thus often represented by a tuple $(S, +, ·, ′, 0, 1)$ which carries all the relevant components.
- A variable $x \in S$ is called a Boolean variable.
- A combination of some elements of $S$ (variables) via the connectives + and · and the complement ′ is a Boolean expression.
- Among the 3 operations “′”, “·” and “+” in a Boolean expression, the “′” operation has the highest precedence, then comes the “·” operation, and then the “+” operation.
Boolean Algebra

- The "0" and "1" in the above are just a notation, two special elements of $S$.
- They are symbols rather than normal numerical values.
- Likewise the operators "+", "·" and "′" are also symbols, each representing the designated special roles.
- A Boolean algebra is thus often represented by a tuple $(S, +, ·, ′, 0, 1)$ which carries all the relevant components.
- A variable $x \in S$ is called a Boolean variable.
- A combination of some elements of $S$ (variables) via the connectives $+$ and $·$ and the complement $′$ is a Boolean expression.
- Among the 3 operations "′", "·" and "+" in a Boolean expression, the "′" operation has the highest precedence, then comes the "·" operation, and then the "+" operation.
  - For example, $a + b′ \cdot c$ in fact means $a + ((b′) \cdot c)$.

▶ Each property in the definition of a Boolean algebra has its dual as part of the definition.
▶ The dual is obtained by interchanging $+$ with $·$ and $0$ with $1$.
▶ Unary operation $′$ remains unchanged.
Boolean Algebra

- The "0" and "1" in the above are just a notation, two special elements of $S$.
- They are symbols rather than normal numerical values.
- Likewise the operators "+", "·" and "′" are also symbols, each representing the designated special roles.
- A Boolean algebra is thus often represented by a tuple $(S, +, ·, ′, 0, 1)$ which carries all the relevant components.
- A variable $x \in S$ is called a Boolean variable.
- A combination of some elements of $S$ (variables) via the connectives + and · and the complement ′ is a Boolean expression.
- Among the 3 operations "′", "·" and "+", in a Boolean expression, the "′" operation has the highest precedence, then comes the "·" operation, and then the "+" operation.
  - For example, $a + b′ \cdot c$ in fact means $a + ((b′) \cdot c)$.
- Each property in the definition of a Boolean algebra has its dual as part of the definition.
Boolean Algebra

- The ”0” and ”1” in the above are just a notation, two special elements of \( S \).
- They are symbols rather than normal numerical values.
- Likewise the operators “+”, “.” and “′” are also symbols, each representing the designated special roles.
- A Boolean algebra is thus often represented by a tuple \((S, +, ·, ′, 0, 1)\) which carries all the relevant components.
- A variable \( x \in S \) is called a Boolean variable.
- A combination of some elements of \( S \) (variables) via the connectives + and · and the complement ′ is a Boolean expression.
- Among the 3 operations “′”, “.” and “+” in a Boolean expression, the “′” operation has the highest precedence, then comes the “.” operation, and then the “+” operation.
  - For example, \( a + b′ \cdot c \) in fact means \( a + ((b′) \cdot c) \).
- Each property in the definition of a Boolean algebra has its dual as part of the definition.
  - The dual is obtained by interchanging + with · and 0 with 1.
Boolean Algebra

- The ”0” and ”1” in the above are just a notation, two special elements of $S$.
- They are symbols rather than normal numerical values.
- Likewise the operators “+”, “·” and “′” are also symbols, each representing the designated special roles.
- A Boolean algebra is thus often represented by a tuple $(S, +, ·, ′, 0, 1)$ which carries all the relevant components.
- A variable $x \in S$ is called a Boolean variable.
- A combination of some elements of $S$ (variables) via the connectives + and · and the complement ′ is a Boolean expression.
- Among the 3 operations “′”, “·” and “+” in a Boolean expression, the “′” operation has the highest precedence, then comes the “·” operation, and then the “+” operation.
  - For example, $a + b′ \cdot c$ in fact means $a + ((b′) \cdot c)$.
- Each property in the definition of a Boolean algebra has its dual as part of the definition.
  - The dual is obtained by interchanging + with · and 0 with 1.
  - Unary operation ′ remains unchanged.
Theorem

Let \((S, +, \cdot, ', 0, 1)\) be a Boolean algebra. Then the following properties hold for all \(a, b, c \in S\)

- **P1.** \(a + a = a; \quad a \cdot a = a.\) \hspace{1cm} \text{(idempotent laws)}
- **P2.** \(a + 1 = 1; \quad a \cdot 0 = 0.\) \hspace{1cm} \text{(dominance laws)}
- **P3.** \((a')' = a.\) \hspace{1cm} \text{(double complement)}
- **P4.** \(a + a \cdot b = a.\) \hspace{1cm} \text{(absorption law)}
- **P5.** If \(a + c = 1, \ a \cdot c = 0,\) then \(c = a'.\) \hspace{1cm} \text{(uniqueness of inverses)}
- **P6.** If \(a \cdot c = b \cdot c, \ a \cdot c' = b \cdot c',\) then \(a = b.\) \hspace{1cm} \text{(cancellation law)}
- **P7.** If \(a + c = b + c, \ a + c' = b + c',\) then \(a = b.\) \hspace{1cm} \text{(cancellation law)}
Proof of the Theorem

Proof.

For P1, the first half is derived from B3–B5 by

\[ a + a \equiv (a + a) \cdot 1 \equiv (a + a) \cdot (a + a') \equiv a + (a \cdot a') \equiv a + 0 \equiv a, \]

where the names on the equality sign indicate the property being used, while the second half is derived by

\[ a \equiv a \cdot 1 \equiv a \cdot (a + a') \equiv a \cdot a + a \cdot a' \equiv a \cdot a + 0 \equiv a \cdot a. \]

For P2, we need to observe

\[ a + 1 \equiv a + (a + a') \equiv (a + a) + a' \equiv a + a' \equiv 1 \]
\[ a \cdot 0 \equiv a \cdot (a \cdot a') \equiv (a \cdot a) \cdot a' \equiv a \cdot a' \equiv 0. \]
Proof of the Theorem

Proof.
The proof of P3 is

\[ a'' \overset{B^4}{=} a'' \cdot 1 \overset{B^5}{=} a'' \cdot (a + a') \overset{B^3}{=} a'' \cdot a + a'' \cdot a' \overset{B^5}{=} a'' \cdot a + 0 \overset{B^5}{=} a'' \cdot a + a' \cdot a \overset{B^3}{=} (a'' + a') \cdot a \overset{B^5}{=} 1 \cdot a \overset{B^4}{=} a. \]

The proof of P4 is

\[ a + a \cdot b \overset{B^4}{=} a \cdot 1 + a \cdot b \overset{B^3}{=} a \cdot (1 + b) \overset{B^1, P^2}{=} a \cdot 1 \overset{B^4}{=} a. \]

The other properties, P5–P7, can be derived similarly.

Note. An immediate corollary of P5 in the above theorem is that \(0' = 1\) and \(1' = 0\) hold on any Boolean algebra \((S, +, \cdot', 0, 1)\).
Example 2

Example

Switching system \((S, +, \cdot, ', 0, 1)\) with \(S = \{0, 1\}\) is a Boolean algebra.
Example 2

Example

Switching system \((S, +, \cdot, ', 0, 1)\) with \(S = \{0, 1\}\) is a Boolean algebra.

Solution. We need to show B1 – B5, by letting "+", "." and "'" specifically denote switches in parallel, in series and in complementation respectively. The identities in B1 are valid because

\[
\begin{align*}
\text{i.e. } a + b &= b + a \\
\text{i.e. } a \cdot b &= b \cdot a
\end{align*}
\]

The proof of B2 – B5 is elementary.
Example 2

Hence we’ll simply show only the first half of B3, i.e.

\[ a + (b \cdot c) = (a + b) \cdot (a + c). \]

\[
\begin{array}{c|ccc|cc|cc}
  a & b & c & b \cdot c & a + b & a + c & a + (b \cdot c) & (a + b) \cdot (a + c) \\
  \hline
  0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
  0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
  0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
  1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
  1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

all corresponding values exactly same
Example 2

- Hence we’ll simply show only the first half of B3, i.e.
  \[ a + (b \cdot c) = (a + b) \cdot (a + c). \]
- We’ll show the identity by the use of the evaluation table below

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>b \cdot c</td>
<td>a + b</td>
<td>a + c</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>0</td>
<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>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</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>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

all corresponding values exactly same
## Example 3

### Example

Let $\mathbb{R}$ be the set of real numbers, $+, \times$ be the normal addition and multiplication, $0, 1 \in \mathbb{R}$ be the normal numbers. If we define $\prime$ by $x' = 1 - x$ for any $x \in \mathbb{R}$, then is $(\mathbb{R}, +, \times, ', 0, 1)$ a Boolean algebra?

## Solution.

No. Because neither the first half of B3 nor the second half of B5 is satisfied. For example, $1 + (2 \times 3) \neq (1 + 2) \times (1 + 3)$.
Example 3

Example

Let \( \mathbb{R} \) be the set of real numbers, \(+, \times\) be the normal addition and multiplication, \(0, 1 \in \mathbb{R}\) be the normal numbers. If we define \( \prime \) by \( x' = 1 - x \) for any \( x \in \mathbb{R} \), then is \((\mathbb{R}, +, \times, ', 0, 1)\) a Boolean algebra?

Solution. No. Because neither the first half of B3 nor the second half of B5 is satisfied. For example,

\[
1 + (2 \times 3) \neq (1 + 2) \times (1 + 3).
\]
Example 4

Example

Suppose $S = \{1, 2, 3, 5, 6, 10, 15, 30\}$. Let $a + b$ denote the least common multiple of $a$ and $b$, $a \cdot b$ denote the greatest common divisor of $a$ and $b$, and $a' = \frac{30}{a}$. Prove $(S, +, \cdot', 1, 30)$ is a Boolean algebra.
Switching systems and symbolic logic are essentially the same.
Algebraic Equivalence

- Switching systems and symbolic logic are essentially the same.
- Furthermore they both form a Boolean algebra \((S, +, \cdot, \cdot', 0, 1)\) on a set \(S = \{0, 1\}\), where

<table>
<thead>
<tr>
<th>Boolean algebra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switching systems</td>
</tr>
<tr>
<td>Symbolic logic</td>
</tr>
<tr>
<td>(S = {0, 1})</td>
</tr>
<tr>
<td>{open, closed}</td>
</tr>
<tr>
<td>{(F), (T)}</td>
</tr>
<tr>
<td>(+)</td>
</tr>
<tr>
<td>(p \lor q) (&quot;or&quot;)</td>
</tr>
<tr>
<td>(\cdot)</td>
</tr>
<tr>
<td>(p \land q) (&quot;and&quot;)</td>
</tr>
<tr>
<td>(\cdot')</td>
</tr>
<tr>
<td>(\sim p) (&quot;not&quot;)</td>
</tr>
<tr>
<td>0 circuit open</td>
</tr>
<tr>
<td>1 circuit closed</td>
</tr>
<tr>
<td>contradiction</td>
</tr>
<tr>
<td>tautology</td>
</tr>
</tbody>
</table>
Algebraic Equivalence

- Switching systems and symbolic logic are essentially the same.
- Furthermore, they both form a Boolean algebra \((S, +, \cdot, \cdot', 0, 1)\) on a set \(S = \{0, 1\}\), where
  - 0 is open in the switching systems and is \(F\) in the symbolic logic, while
Switching systems and symbolic logic are essentially the same. Furthermore they both form a Boolean algebra \((S, +, \cdot, ', 0, 1)\) on a set \(S = \{0, 1\}\), where

- 0 is open in the switching systems and is \(F\) in the symbolic logic, while
- 1 is closed in the switching systems and is \(T\) in the symbolic logic.

The correspondence can be seen in the following table:

<table>
<thead>
<tr>
<th>Boolean algebra</th>
<th>Switching systems</th>
<th>Symbolic logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S = {0, 1})</td>
<td>{open, closed}</td>
<td>{(F, T)}</td>
</tr>
<tr>
<td>(+)</td>
<td>(p \lor q) (&quot;or&quot;)</td>
<td>(p \lor q)</td>
</tr>
<tr>
<td>(\cdot)</td>
<td>(p \land q) (&quot;and&quot;)</td>
<td>(p \land q)</td>
</tr>
<tr>
<td>(')</td>
<td>(\sim p) (&quot;not&quot;)</td>
<td>(\sim p)</td>
</tr>
</tbody>
</table>

\(0\) circuit open \(\rightarrow\) contradiction
\(1\) circuit closed \(\rightarrow\) tautology
Algebraic Equivalence

- Switching systems and symbolic logic are essentially the same.
- Furthermore they both form a Boolean algebra \((S, +, \cdot, \cdot', 0, 1)\) on a set \(S = \{0, 1\}\), where
  - 0 is open in the switching systems and is \(F\) in the symbolic logic, while
  - 1 is closed in the switching systems and is \(T\) in the symbolic logic.
- The correspondence can be seen in the following table
Algebraic Equivalence

- Switching systems and symbolic logic are essentially the same.
- Furthermore they both form a Boolean algebra \((S, +, \cdot, \cdot', 0, 1)\) on a set \(S = \{0, 1\}\), where
  - 0 is open in the switching systems and is \(F\) in the symbolic logic, while
  - 1 is closed in the switching systems and is \(T\) in the symbolic logic.
- The correspondence can be seen in the following table
Algebraic Equivalence

Switching systems and symbolic logic are essentially the same.

Furthermore they both form a Boolean algebra \((S, +, \bullet, ', 0, 1)\) on a set \(S = \{0, 1\}\), where

- 0 is open in the switching systems and is \(F\) in the symbolic logic, while
- 1 is closed in the switching systems and is \(T\) in the symbolic logic.

The correspondence can be seen in the following table

<table>
<thead>
<tr>
<th>Boolean algebra</th>
<th>Switching systems</th>
<th>Symbolic logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S = {0, 1})</td>
<td>{open, closed}</td>
<td>({F, T})</td>
</tr>
<tr>
<td>+</td>
<td>(x + y) (in parallel)</td>
<td>(p \lor q) (“or”)</td>
</tr>
<tr>
<td>(\bullet)</td>
<td>(x \cdot y) (in series)</td>
<td>(p \land q) (“and”)</td>
</tr>
<tr>
<td>,</td>
<td>(x') (complement)</td>
<td>(\sim p) (“not”)</td>
</tr>
<tr>
<td>0</td>
<td>circuit open</td>
<td>contradiction</td>
</tr>
<tr>
<td>1</td>
<td>circuit closed</td>
<td>tautology</td>
</tr>
</tbody>
</table>
Algebraic Equivalence

Hence properties B1–B5 and P1–P7 will also hold in symbolic logic, when “+”, “⋅”, “′”, “0” and “1” are replaced by “∨”, “∧”, “∼”, contradiction and tautology respectively.

Hence, for example,

\[ a \lor (b \land c) \equiv (a \lor b) \land (a \lor c) \]
\[ a \land (b \lor c) \equiv (a \land b) \lor (a \land c) \]
\[ a \lor a \equiv a, \quad a \land a \equiv a, \quad \neg \neg a \equiv a, \quad a \lor \bot \equiv a, \quad a \land \top \equiv a \]

hold for any propositions \( a \), \( b \) and \( c \), where \( \bot \) represents a contradiction and \( \top \) represents a tautology.
Algebraic Equivalence

Hence properties B1–B5 and P1–P7 will also hold in symbolic logic, when “+”, “·”, “′”, “0” and “1” are replaced by “∨”, “∧”, “∼”, contradiction and tautology respectively.

- Hence, for example,
Algebraic Equivalence

Hence properties B1–B5 and P1–P7 will also hold in symbolic logic, when “+”, “.”, “⊤”, “0” and “1” are replaced by “∨”, “∧”, “∼”, contradiction and tautology respectively.

- Hence, for example,
Algebraic Equivalence

Hence properties B1–B5 and P1–P7 will also hold in symbolic logic, when “+”, “·”, “′”, “0” and “1” are replaced by “∨”, “∧”, “∼”, contradiction and tautology respectively.

Hence, for example,

\[ a \lor (b \land c) \equiv (a \lor b) \land (a \lor c), \quad a \land (b \lor c) \equiv (a \land b) \lor (a \land c), \]
Algebraic Equivalence

Hence properties B1–B5 and P1–P7 will also hold in symbolic logic, when “+”, “·”, “′”, “0” and “1” are replaced by “∨”, “∧”, “∼”, contradiction and tautology respectively.

Hence, for example,

\[ a \lor (b \land c) \equiv (a \lor b) \land (a \lor c), \quad a \land (b \lor c) \equiv (a \land b) \lor (a \land c), \]

\[ a \lor a \equiv a, \quad a \land a \equiv a, \quad \sim (\sim a) \equiv a, \quad a \lor \bot \equiv a, \quad a \land \top \equiv a, \]
Hence properties B1–B5 and P1–P7 will also hold in symbolic logic, when “+”, “·”, “′”, “0” and “1” are replaced by “∨”, “∧”, “∼”, contradiction and tautology respectively.

Hence, for example,

\[ a \lor (b \land c) \equiv (a \lor b) \land (a \lor c), \quad a \land (b \lor c) \equiv (a \land b) \lor (a \land c), \]

\[ a \lor a \equiv a, \quad a \land a \equiv a, \quad \sim (\sim a) \equiv a, \quad a \lor \bot \equiv a, \quad a \land \top \equiv a, \]

\[ a \lor (a \land b) \equiv a, \quad a \land (\sim a) \equiv \bot, \quad a \lor (\sim a) \equiv \top, \quad a \land \bot \equiv \bot, \quad a \lor \top \equiv \top \]

hold for any propositions \( a, b \) and \( c \), where \( \bot \) represents a contradiction and \( \top \) represents a tautology.
Example 5

Example

Convert \((p \lor q) \rightarrow r\) into the corresponding Boolean expression.
Example 5

Example

Convert \((p \lor q) \rightarrow r\) into the corresponding Boolean expression.

Solution. Since \(p \rightarrow q\) is equivalent to \((\sim p) \lor q\), we see that \((p \lor q) \rightarrow r\) is equivalent to \((\sim(p \lor q)) \lor r\) which is thus converted to \((p + q)' + r\).
Example 6 (De Morgan’s Laws)

Example

Let \((S, +, \cdot, ', 0, 1)\) be a Boolean algebra, then for any \(x, y \in S\)

\[(x + y)' = x' \cdot y',\]

\[(x \cdot y)' = x' + y'.\]
Example 6 (De Morgan’s Laws)

Example
Let \((S, +, \cdot, ' , 0, 1)\) be a Boolean algebra, then for any \(x, y \in S\)

\[(x + y)' = x' \cdot y',\]
\[(x \cdot y)' = x' + y'.\]

Solution. Proof obvious from the theorem in the previous section.
Example 7

Example

Suppose \((\mathbb{T}, \spadesuit, \diamondsuit, \heartsuit, \sqcup, \sqcap)\) is a Boolean algebra. Show \((\sqcap \heartsuit) \spadesuit \sqcup = \sqcap\).
Example 7

Example

Suppose \((\mathbb{T}, \blacklozenge, \blacklozenge, \odot, \sqcup, \sqcap)\) is a Boolean algebra. Show \((\sqcup \odot) \blacklozenge \sqcup = \sqcap\).

Solution. Recall that when we say \((\mathbb{T}, \blacklozenge, \blacklozenge, \odot, \sqcup, \sqcap)\) is a Boolean algebra, the tuple means that

\(\blacklozenge, \blacklozenge\) and \(\odot\) correspond respectively to the

“+”, “.” and “\(\prime\)” operations entailed by a Boolean algebra, and that

\(\sqcup\) and \(\sqcap\) correspond respectively to the

“0” and “1” elements possessed by the Boolean algebra.

Hence \((\sqcup \odot) \blacklozenge \sqcup = \sqcap\) is the same as \((0\prime) + 0 = 1\) and is thus obviously true.
Example 8

Example

Let $B = \{0, 1\}$ and $(B, +, \cdot, ', 0, 1)$ be a Boolean algebra.
Let $B^n$ denote the set of all the tuples $(x_1, x_2, ..., x_n)$ with $x_1, ..., x_n$ being any elements of $B$, i.e.

$$B^n \overset{\text{def}}{=} \{(x_1, ..., x_n) \mid x_1 \in B, ..., x_n \in B\}$$

Then $(B^n, +, \cdot, ', 0, 1)$ is also a Boolean algebra if

$$0 = (0, \cdots, 0),$$
$$1 = (1, \cdots, 1),$$
$$x + y = (x_1 + y_1, \cdots, x_n + y_n),$$
$$x \cdot y = (x_1 \cdot y_1, \cdots, x_n \cdot y_n),$$
$$x' = (x'_1, \cdots, x'_n).$$
Sets connection with Boolean Algebra

- If we make the correspondence between

\[ \emptyset, U, \cup, \cap, ' \quad \text{and} \quad 0, 1, +, \cdot, ' \]

for sets and Boolean algebra respectively, we see that properties S1–S5 are exactly those B1–B5 for the definition of Boolean algebra.
Sets connection with Boolean Algebra

- If we make the correspondence between
  \[ \emptyset, U, \cup, \cap, ' \text{ and } 0, 1, +, \cdot, ' \]
  for sets and Boolean algebra respectively, we see that properties S1–S5 are exactly those B1–B5 for the definition of Boolean algebra.
- Hence for any nonempty set \( S \), for instance, \( (\mathcal{P}(S), \cup, \cap, ', \emptyset, S) \) is a Boolean algebra.
Sets connection with Boolean Algebra

- If we make the correspondence between
  \[ \emptyset, U, \cup, \cap, ' \] and \[ 0, 1, +, \cdot, ' \]
  for sets and Boolean algebra respectively, we see that properties S1–S5
  are exactly those B1–B5 for the definition of Boolean algebra.
- Hence for any nonempty set \( S \), for instance, \((\mathcal{P}(S), \cup, \cap, ' , \emptyset, S)\) is a
  Boolean algebra.
Sets connection with Boolean Algebra

If we make the correspondence between

\[ \emptyset, U, \cup, \cap, ', \text{ and } 0, 1, +, \cdot, '\]

for sets and Boolean algebra respectively, we see that properties S1–S5 are exactly those B1–B5 for the definition of Boolean algebra.

Hence for any nonempty set \( S \), for instance, \( (\mathcal{P}(S), \cup, \cap, ', \emptyset, S) \) is a Boolean algebra.

**Theorem**

If \( B \) is a Boolean algebra with exactly \( n \) elements then \( n = 2^m \) for some \( m \). Furthermore \( B \) and \( (\mathcal{P}\{1, 2, \cdots m\}, \cup, \cap, l, \emptyset, \{1, 2, \cdots , m\}) \) essentially represent the same Boolean algebra.
Sets connection with Boolean Algebra

- If we make the correspondence between
  
  \( \emptyset, \, U, \, \cup, \, \cap, \, ' \) and \( 0, \, 1, \, +, \, \cdot, \, ' \)

  for sets and Boolean algebra respectively, we see that properties S1–S5
  are exactly those B1–B5 for the definition of Boolean algebra.
- Hence for any nonempty set \( S \), for instance, \((\mathcal{P}(S), \cup, \cap, ', \emptyset, S)\) is a
  Boolean algebra.

**Theorem**

If \( B \) is a Boolean algebra with exactly \( n \) elements then \( n = 2^m \) for some \( m \).
Furthermore \( B \) and \((\mathcal{P}(\{1, 2, \cdots m\}), \cup, \cap, I, \emptyset, \{1, 2, \cdots, m\})\) essentially
represent the same Boolean algebra.

**Example**

9. Let \( U = \{2, 3\} \) and \( S = \mathcal{P}(U) \). Then \((S, \cup, \cap, ', \emptyset, U)\) is a Boolean algebra.
   Notice that the set \( S \) in this case contains more than 2 elements.