Discrete Mathematics and its Applications

Ngày 12 tháng 9 năm 2011
(Introduction)

“The universe cannot be read until we have learnt the language and become familiar with the characters in which it is written. It is written in mathematical language... without which means it is humanly impossible to comprehend a single word"

Logic is the bridge between the language the universe is written in and mathematics.

In this lecture we shall learn the basic entities of logic:
(Introduction)

“The universe cannot be read until we have learnt the language and become familiar with the characters in which it is written. It is written in mathematical language... without which means it is humanly impossible to comprehend a single word"

Logic is the bridge between the language the universe is written in and mathematics.

In this lecture we shall learn the basic entities of logic:

1. Propositions
(Introduction)

“The universe cannot be read until we have learnt the language and become familiar with the characters in which it is written. It is written in mathematical language... without which means it is humanly impossible to comprehend a single word"

Logic is the bridge between the language the universe is written in and mathematics.

In this lecture we shall learn the basic entities of logic:

1. Propositions
2. Boolean Variables
(Introduction)

“The universe cannot be read until we have learnt the language and become familiar with the characters in which it is written. It is written in mathematical language... without which means it is humanly impossible to comprehend a single word"

Logic is the bridge between the language the universe is written in and mathematics.

In this lecture we shall learn the basic entities of logic:

1. Propositions
2. Boolean Variables
3. Logical (boolean) operators.
(Introduction)

“The universe cannot be read until we have learnt the language and become familiar with the characters in which it is written. It is written in mathematical language... without which means it is humanly impossible to comprehend a single word"

Logic is the bridge between the language the universe is written in and mathematics.

In this lecture we shall learn the basic entities of logic:

1. Propositions
2. Boolean Variables
3. Logical (boolean) operators.
4. Truth tables.
Definition

A **Proposition** is a statement of a fact which is either true or false but not both.

1. Today is Saturday
2. It is raining today.
3. If $n$ is an integer then $(2n+1)^2 \mod 8 = 1$.
4. If $n$ is an odd prime number then $2^n - 1 \mod n = 1$.
5. There are no positive integers $x$, $y$, $z$ satisfying the equality $x^5 + y^5 = z^5$.
6. There are infinitely many prime numbers $q$ such that $q = 4p + 1$ where $p$ is prime.
Propositions

Definition

A **Proposition** is a statement of a fact which is either true or false but not both.

Example
Propositions

Definition

A **Proposition** is a statement of a fact which is either true or false but not both.

Example

1. *Today is Saturday*
Propositions

Definition

A **Proposition** is a statement of a fact which is either true or false but not both.

Example

1. *Today is Saturday*
2. *It is raining today.*
Propositions

Definition

A **Proposition** is a statement of a fact which is either true or false but not both.

Example

1. *Today is Saturday*
2. *It is raining today.*
3. *if n is an integer then* $(2n + 1)^2 \mod 8 = 1$. 
Propositions

Definition

A **Proposition** is a statement of a fact which is either true or false but not both.

Example

1. *Today is Saturday*
2. *It is raining today.*
3. *if n is an integer then* \((2n + 1)^2 \mod 8 = 1\).*
4. *if n is an odd prime number then* \(2^{n-1} \mod n = 1\).*

There are no positive integers \(x, y, z\) satisfying the equality \(x^5 + y^5 = z^5\).

There are infinitely many prime numbers \(q\) such that \(q = 4p + 1\) where \(p\) is prime.
Propositions

Definition

A **Proposition** is a statement of a fact which is either true or false but not both.

Example

1. *Today is Saturday*
2. *It is raining today.*
3. *if n is an integer then \((2n + 1)^2 \mod 8 = 1.*
4. *if n is an odd prime number then \(2^{n-1} \mod n = 1.*
5. *There are no positive integers \(x, y, z\) satisfying the equality \(x^5 + y^5 = z^5 \)
Propositions

Definition

A **Proposition** is a statement of a fact which is either true or false but not both.

Example

1. *Today is Saturday*
2. *It is raining today.*
3. *if n is an integer then* \((2n + 1)^2 \mod 8 = 1\).*
4. *if n is an odd prime number then* \(2^{n-1} \mod n = 1\).
5. *There are no positive integers* \(x, y, z\) *satisfying the equality* \(x^5 + y^5 = z^5\).
6. *There are infinitely many prime numbers* \(q\) *such that* \(q = 4p + 1\) *where* \(p\) *is prime.*
Here are some examples of sentences which are not propositions.

Example
Non-propositions

Here are some examples of sentences which are not propositions.

Example

1. *Do not drive over the speed limit.*
Non-propositions

Here are some examples of sentences which are not propositions.

Example

1. *Do not drive over the speed limit.*
2. *Do not use your cell phone while driving.*
Non-propositions

Here are some examples of sentences which are not propositions.

Example

1. *Do not drive over the speed limit.*
2. *Do not use your cell phone while driving.*
3. \( a^2 + b^2 = c^2 \)
Non-propositions

Here are some examples of sentences which are not propositions.

Example

1. *Do not drive over the speed limit.*
2. *Do not use your cell phone while driving.*
3. \( a^2 + b^2 = c^2 \)
4. *Hoang lives in Hanoi. He owns a xe may. Hoang fixes all xe mays in Hanoi belonging to people that do not fix their own xe may.*
Here are some examples of sentences which are not propositions.

**Example**

1. *Do not drive over the speed limit.*
2. *Do not use your cell phone while driving.*
3. \(a^2 + b^2 = c^2\)
4. *Hoang lives in Hanoi. He owns a xe may. Hoang fixes all xe mays in Hanoi belonging to people that do not fix their own xe may.*
Non-propositions

Here are some examples of sentences which are not propositions.

Example

1. *Do not drive over the speed limit.*
2. *Do not use your cell phone while driving.*
3. $a^2 + b^2 = c^2$
4. *Hoang lives in Hanoi. He owns a xe may. Hoang fixes all xe mays in Hanoi belonging to people that do not fix their own xe may.*

- 1 and 2 are not propositions as they do not state a fact.
Non-propositions

Here are some examples of sentences which are not propositions.

Example

1. Do not drive over the speed limit.
2. Do not use your cell phone while driving.
3. \( a^2 + b^2 = c^2 \)
4. Hoang lives in Hanoi. He owns a xe may. Hoang fixes all xe mays in Hanoi belonging to people that do not fix their own xe may.

- 1 and 2 are not propositions as they do not state a fact.
- 3 can be both true and false, depending on the values of \( a, b, c \).
Non-propositions

Here are some examples of sentences which are not propositions.

Example

1. *Do not drive over the speed limit.*
2. *Do not use your cell phone while driving.*
3. \[a^2 + b^2 = c^2\]
4. *Hoang lives in Hanoi. He owns a xe may. Hoang fixes all xe mays in Hanoi belonging to people that do not fix their own xe may.*

- 1 and 2 are not propositions as they do not state a fact.
- 3 can be both true and false, depending on the values of \(a, b, c\).
- 4 is a bit more intricate. Hoang cannot fix his own xe may since he fixes only those belonging to people that do not fix their own xe may but if he does not fix his own xe may then he is fixing it.
Logic operations and compound propositions were introduced by the English mathematician George Boole in 1848. This lay the foundation of developing the digital computer 100 years later.
Logic operations and compound propositions were introduced by the English mathematician George Boole in 1848. This lay the foundation of developing the digital computer 100 years later.

We shall denote propositions by letters: \( p, q, r, s, \ldots \).

**Definition**

A **Boolean Variable** is a variable whose domain is propositions and range just two values, commonly denoted by **TRUE, FALSE**. Sometimes we also use \( \{1, 2\} \).
Logic operations and compound propositions were introduced by the English mathematician George Boole in 1848. This lay the foundation of developing the digital computer 100 years later.

We shall denote propositions by letters: $p, q, r, s, \ldots$.

**Definition**

A **Boolean Variable** is a variable whose domain is propositions and range just two values, commonly denoted by **TRUE, FALSE**. Sometimes we also use $\{1, 2\}$.

**Comment**

Almost all programming languages include boolean variables.
Question

What can be done with a single boolean variable that has only two values?
Logic operations

Question

*What can be done with a single boolean variable that has only two values?*

Answer

*Not much more than a light switch, it can be off or on. But combining an array of boolean variables, like 32 in common processors yields $2^{32}$ different patterns, That is more than 4,000,000,000 patterns!*
Question
What can be done with a single boolean variable that has only two values?

Answer
Not much more than a light switch, it can be off or on. But combining an array of boolean variables, like 32 in common processors yields $2^{32}$ different patterns, That is more than $4,000,000,000$ patterns!

Combining boolean variable is done with logic or boolean operators.
Example (Compound Propositions)

1. Phuong's PC does not run UNIX.

2. The speed limit in Hanoi for xe may is 40 km/hour and for trucks 30 km/hour.

3. If $n$ is prime then if $a < n$ then $a^n - 1 \mod n = 1$.

4. If $n$ is prime or $n = p^k$, $p$ prime then there is a finite field of order $n$. 

Discussion

Each proposition is composed of one or more propositions connected by key words:

- **not**: number 1 has one proposition.
- **and**: number 2 has two propositions.
- **or**: number 4 has 3 propositions.
- **if then**: number 3 has 3 propositions.
Example (Compound Propositions)

1. Phuong’s PC does not run UNIX.
Example (Compound Propositions)

1. Phuong’s PC does not run UNIX.
2. The speed limit in Hanoi for xe may is 40 km/hour and for trucks 30 km/hour.
Example (Compound Propositions)

1. Phuong’s PC does not run UNIX.
2. The speed limit in Hanoi for xe may is 40 km/hour and for trucks 30 km/hour.
3. If $n$ is prime then if $a < n$ then $a^{n-1}$ mod $n = 1$
Example (Compound Propositions)

1. Phuong’s PC does not run UNIX.
2. The speed limit in Hanoi for xe may is 40 km/hour and for trucks 30 km/hour.
3. If $n$ is prime then if $a < n$ then $a^{n-1} \mod n = 1$
4. if $n$ is prime or $n = p^k$, $p$ prime then there is a finite field of order $n$. 
Example (Compound Propositions)

1. Phuong’s PC does **not** run UNIX.
2. The speed limit in Hanoi for xe may is 40 km/hour **and** for trucks 30 km/hour.
3. If \( n \) is prime then if \( a < n \) then \( a^{n-1} \mod n = 1 \)
4. If \( n \) is prime or \( n = p^k \), \( p \) prime then there is a finite field of order \( n \).

Discussion

*Each proposition is composed of one or more propositions connected by key words:*
Example (Compound Propositions)

1. Phuong’s PC does not run UNIX.
2. The speed limit in Hanoi for xe may is 40 km/hour and for trucks 30 km/hour.
3. If $n$ is prime then if $a < n$ then $a^{n-1} \mod n = 1$
4. if $n$ is prime or $n = p^k$, $p$ prime then there is a finite field of order $n$.

Discussion

Each proposition is composed of one or more propositions connected by key words:

1. not: number 1 has one proposition.
Example (Compound Propositions)

1. Phuong’s PC does not run UNIX.
2. The speed limit in Hanoi for xe may is 40 km/hour and for trucks 30 km/hour.
3. If $n$ is prime then if $a < n$ then $a^{n-1} \mod n = 1$
4. if $n$ is prime or $n = p^k$, $p$ prime then there is a finite field of order $n$.

Discussion

Each proposition is composed of one or more propositions connected by key words:

1. **not**: number 1 has one proposition.
2. **and**: number 2 has two propositions
Example (Compound Propositions)

1. Phuong’s PC does **not** run UNIX.
2. The speed limit in Hanoi for xe may is 40 km/hour **and** for trucks 30 km/hour.
3. If \( n \) is prime then if \( a < n \) then \( a^{n-1} \mod n = 1 \)
4. If \( n \) is prime or \( n = p^k, p \) prime then there is a finite field of order \( n \).

Discussion

Each proposition is composed of one or more propositions connected by key words:

1. **not**: number 1 has one proposition.
2. **and**: number 2 has two propositions
3. **or**: number 4 has 3 propositions
Example (Compound Propositions)

1. Phuong’s PC does not run UNIX.
2. The speed limit in Hanoi for xe may is 40 km/hour and for trucks 30 km/hour.
3. If \( n \) is prime then if \( a < n \) then \( a^{n-1} \mod n = 1 \)
4. if \( n \) is prime or \( n = p^k \), \( p \) prime then there is a finite field of order \( n \).

Discussion

Each proposition is composed of one or more propositions connected by key words:

1. not: number 1 has one proposition.
2. and: number 2 has two propositions
3. or: number 4 has 3 propositions
4. if then number 3 has 3 propositions.
**not** is a unary operator, \( \neg p \) inverts the truth value of \( p \).
1. **not** is a unary operator, $\neg p$ inverts the truth value of $p$.

2. **and** is a binary operator, $p \land q$ is **true** only when both $p$ and $q$ are true.
not is a unary operator, \( \neg p \) inverts the truth value of \( p \).

and is a binary operator, \( p \land q \) is true only when both \( p \) and \( q \) are true.

or is a binary operator, \( p \lor q \) is false only when both are false.
1. **not** is a unary operator, $\neg p$ inverts the truth value of $p$.
2. **and** is a binary operator, $p \land q$ is **true** only when both $p$ and $q$ are true.
3. **or** is a binary operator, $p \lor q$ is **false** only when both are false.
4. **implies** (if then) is a binary operator $p \rightarrow q$ is **false** only when $p$ is true and $q$ is false.
1. **not** is a unary operator, $\neg p$ inverts the truth value of $p$.

2. **and** is a binary operator, $p \land q$ is **true** only when both $p$ and $q$ are true.

3. **or** is a binary operator, $p \lor q$ is **false** only when both are false.

4. **implies** (if then) is a binary operator $p \rightarrow q$ is **false** only when $p$ is true and $q$ is false.

**Truth tables** are used to describe these and also the truth values of compound propositions.
1. **not** is a unary operator, $\neg p$ inverts the truth value of $p$.

2. **and** is a binary operator, $p \land q$ is **true** only when both $p$ and $q$ are true.

3. **or** is a binary operator, $p \lor q$ is **false** only when both are false.

4. **implies** (if then) is a binary operator $p \rightarrow q$ is **false** only when $p$ is true and $q$ is false.

**Truth tables** are used to describe these and also the truth values of compound propositions.

All these operators can be physically implemented electronically.
1. **not** is a unary operator, \( \neg p \) inverts the truth value of \( p \).
2. **and** is a binary operator, \( p \land q \) is **true** only when both \( p \) and \( q \) are true.
3. **or** is a binary operator, \( p \lor q \) is **false** only when both are false.
4. **implies** (if then) is a binary operator \( p \rightarrow q \) is **false** only when \( p \) is true and \( q \) is false.

**Truth tables** are used to describe these and also the truth values of compound propositions.

1. All these operators can be physically implemented electronically.
2. These are the building blocks of micro-processors and many other systems.
1. **not** is a unary operator, \( \neg p \) inverts the truth value of \( p \).
2. **and** is a binary operator, \( p \land q \) is **true** only when both \( p \) and \( q \) are true.
3. **or** is a binary operator, \( p \lor q \) is **false** only when both are false.
4. **implies** (if then) is a binary operator \( p \rightarrow q \) is **false** only when \( p \) is true and \( q \) is false.

**Truth tables** are used to describe these and also the truth values of compound propositions.

1. All these operators can be physically implemented electronically.
2. These are the building blocks of micro-processors and many other systems.
1. **not** is a unary operator, \( \neg p \) inverts the truth value of \( p \).
2. **and** is a binary operator, \( p \land q \) is **true** only when both \( p \) and \( q \) are true.
3. **or** is a binary operator, \( p \lor q \) is **false** only when both are false.
4. **implies** (if then) is a binary operator \( p \rightarrow q \) is **false** only when \( p \) is true and \( q \) is false.

**Truth tables** are used to describe these and also the truth values of compound propositions.

1. All these operators can be physically implemented electronically.
2. These are the building blocks of micro-processors and many other systems.

There are other binary operators. Truth tables will help us understand how to construct them.
Truth Tables

Truth table for the unary operator **not**:

<table>
<thead>
<tr>
<th>p</th>
<th>( \neg p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
Truth Tables

Truth table for the unary operator **not**:

\[
\begin{array}{c|c}
 p & \neg p \\
\hline
 T & F \\
 F & T \\
\end{array}
\]

Truth tables for the binary operators \(\land\ \lor\ \rightarrow\):

\[
\begin{array}{c|c|c|c|c}
 p & q & p \land q & p \lor q & p \rightarrow q \\
\hline
 T & T & T & T & T \\
 F & T & F & T & T \\
 T & F & F & T & F \\
 F & F & F & F & T \\
\end{array}
\]
Evaluating compound propositions with truth tables

Example

*We wish to build the truth table for the compound proposition:*

\[(p \rightarrow q) \land (\neg p \rightarrow q)\]

<table>
<thead>
<tr>
<th>(p)</th>
<th>(q)</th>
<th>(p \rightarrow q)</th>
<th>(\neg p \rightarrow q)</th>
<th>((p \rightarrow q) \land (\neg p \rightarrow q))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T)</td>
<td>(T)</td>
<td>(T)</td>
<td>(T)</td>
<td>(T)</td>
</tr>
<tr>
<td>(F)</td>
<td>(T)</td>
<td>(T)</td>
<td>(T)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(F)</td>
<td>(T)</td>
<td>(F)</td>
</tr>
<tr>
<td>(F)</td>
<td>(F)</td>
<td>(T)</td>
<td>(F)</td>
<td>(F)</td>
</tr>
</tbody>
</table>
Question

How many possible binary operators are there?

How many non-trivial binary operators are there?

How many rows will be needed in the truth table of a compound proposition with 5 different boolean variables?

Comment

Here is a list of commonly used binary operators, their names and description:

1. nor, the reverse of or, $p \downarrow q$ is true only when both $p$ and $q$ are false.
2. nand, the reverse of and, $p \mid q$ is false only when both $p$ and $q$ are true.
3. xor (exclusive or) $p \oplus q$ is true only when they are different (one is true and the other is false).
4. implies $p \rightarrow q$ is false only when $p = true$ and $q = false$.
5. Biconditional $p \leftrightarrow q$ is true only if both are equal.

It should be easy now to construct the truth tables of these binary logic operators.
Question

1. How many possible binary operators are there?

Comment

Here is a list of commonly used binary operators, their names and description:

1. nor, the reverse of or, $p \downarrow q$ is true only when both $p$ and $q$ are false.
2. nand, the reverse of and, $p \mid q$ is false only when both $p$ and $q$ are true.
3. xor (exclusive or), $p \oplus q$ is true only when they are different (one is true and the other is false).
4. implies, $p \rightarrow q$ is false only when $p = true$ and $q = false$.
5. Biconditional, $p \leftrightarrow q$ is true only if both are equal.

It should be easy now to construct the truth tables of these binary logic operators.
Question

1. How many possible binary operators are there?
2. How many non-trivial binary operators are there?

Comment

Here is a list of commonly used binary operators, their names and description:

1. nor, the reverse of or, $p \downarrow q$ is true only when both $p$ and $q$ are false.
2. nand, the reverse of and, $p \mid q$ is false only when both $p$ and $q$ are true.
3. xor (exclusive or), $p \oplus q$ is true only when they are different (one is true and the other is false).
4. implies, $p \rightarrow q$ is false only when $p = \text{true}$ and $q = \text{false}$.
5. Biconditional, $p \leftrightarrow q$ is true only if both are equal.

It should be easy now to construct the truth tables of these binary logic operators.
Question

1. How many possible binary operators are there?
2. How many non-trivial binary operators are there?
3. How many rows will be needed in the truth table of a compound proposition with 5 different boolean variables?
Question

1. How many possible binary operators are there?
2. How many non-trivial binary operators are there?
3. How many rows will be needed in the truth table of a compound proposition with 5 different boolean variables?

Comment

Here is a list of commonly used binary operators, their names and description:

1. **nor**, the reverse of **or**, $p \downarrow q$ is true only when both $p$ and $q$ are false.
2. **nand**, the reverse of **and**, $p \mid q$ is false only when both $p$ and $q$ are true.
3. **xor** (exclusive or), $p \oplus q$ is true only when they are different (one is true and the other is false).
4. **implies**, $p \rightarrow q$ is false only when $p = true$ and $q = false$.
5. **Biconditional**, $p \leftrightarrow q$ is true only if both are equal.

It should be easy now to construct the truth tables of these binary logic operators.
Question

1. How many possible binary operators are there?
2. How many non-trivial binary operators are there?
3. How many rows will be needed in the truth table of a compound proposition with 5 different boolean variables?

Comment

Here is a list of commonly used binary operators, their names and description:

1. nor, the reverse of or, $p \downarrow q$ is true only when both $p$ and $q$ are false.
Question

1. How many possible binary operators are there?
2. How many non-trivial binary operators are there?
3. How many rows will be needed in the truth table of a compound proposition with 5 different boolean variables?

Comment

Here is a list of commonly used binary operators, their names and description:

1. nor, the reverse of or, \( p \downarrow q \) is true only when both \( p \) and \( q \) are false.
2. nand, the reverse of and. \( p \mid q \) is false only when both \( p \) and \( q \) are true.
Question

1. How many possible binary operators are there?
2. How many non-trivial binary operators are there?
3. How many rows will be needed in the truth table of a compound proposition with 5 different boolean variables?

Comment

Here is a list of commonly used binary operators, their names and description:

1. **nor**, the reverse of **or**. \( p \downarrow q \) is true only when both \( p \) and \( q \) are false.
2. **nand**, the reverse of **and**. \( p \mid q \) is false only when both \( p \) and \( q \) are true.
3. **xor** (exclusive or). \( p \oplus q \) is true only when they are different (one is true and the other is false).
Question
1. How many possible binary operators are there?
2. How many non-trivial binary operators are there?
3. How many rows will be needed in the truth table of a compound proposition with 5 different boolean variables?

Comment
Here is a list of commonly used binary operators, their names and description:

1. nor, the reverse of or, $p \downarrow q$ is true only when both $p$ and $q$ are false.
2. nand, the reverse of and. $p \mid q$ is false only when both $p$ and $q$ are true.
3. xor (exclusive or) $p \oplus q$ is true only when they are different (one is true and the other is false).
4. implies $p \rightarrow q$ is false only when $p = true$ and $q = false$. 
Question

1. How many possible binary operators are there?
2. How many non-trivial binary operators are there?
3. How many rows will be needed in the truth table of a compound proposition with 5 different boolean variables?

Comment

Here is a list of commonly used binary operators, their names and description:

1. **nor**, the reverse of **or**. \( p \downarrow q \) is true only when both \( p \) and \( q \) are false.
2. **nand**, the reverse of **and**. \( p \mid q \) is false only when both \( p \) and \( q \) are true.
3. **xor** (exclusive or). \( p \oplus q \) is true only when they are different (one is true and the other is false).
4. **implies** \( p \rightarrow q \) is false only when \( p = \text{true} \) and \( q = \text{false} \).
Question

1. How many possible binary operators are there?
2. How many non-trivial binary operators are there?
3. How many rows will be needed in the truth table of a compound proposition with 5 different boolean variables?

Comment

Here is a list of commonly used binary operators, their names and description:

1. nor, the reverse of or, $p \downarrow q$ is true only when both $p$ and $q$ are false.
2. nand, the reverse of and. $p \mid q$ is false only when both $p$ and $q$ are true.
3. xor (exclusive or) $p \oplus q$ is true only when they are different (one is true and the other is false).
4. implies $p \rightarrow q$ is false only when $p = true$ and $q = false$. 
Question

1. How many possible binary operators are there?
2. How many non-trivial binary operators are there?
3. How many rows will be needed in the truth table of a compound proposition with 5 different boolean variables?

Comment

Here is a list of commonly used binary operators, their names and description:

1. **nor**, the reverse of **or**, \( p \downarrow q \) is true only when both \( p \) and \( q \) are false.
2. **nand**, the reverse of **and**. \( p | q \) is false only when both \( p \) and \( q \) are true.
3. **xor** (exclusive or) \( p \oplus q \) is true only when they are different (one is true and the other is false).
4. **implies** \( p \rightarrow q \) is false only when \( p = true \) and \( q = false \).
Applying multiple logic operations is similar to using arithmetic operations. We need precedence rules. To understand why consider the expressions $p \lor q \land s$. 

<table>
<thead>
<tr>
<th>Operator</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>¬</td>
<td>1</td>
</tr>
<tr>
<td>∧</td>
<td>2</td>
</tr>
<tr>
<td>∨</td>
<td>3</td>
</tr>
<tr>
<td>→</td>
<td>4</td>
</tr>
<tr>
<td>(</td>
<td>-10</td>
</tr>
</tbody>
</table>
Applying multiple logic operations is similar to using arithmetic operations. We need precedence rules. To understand why consider the expressions \( p \lor q \land s \).

Now suppose both \( p \) and \( q \) are true and \( s \) is false. The truth value of this expression will be true if we first evaluate \( p \land s \). But if we first calculate \( p \lor q \) the result is false. So we need precedences. Here they are:
Applying multiple logic operations is similar to using arithmetic operations. We need precedence rules. To understand why consider the expressions $p \lor q \land s$.
Now suppose both $p$ and $q$ are true and $s$ is false. The truth value of this expression will be true if we first evaluate $p \land s$. But if we first calculate $p \lor q$ the result is false. So we need precedences. Here they are:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\neg$</td>
<td>1</td>
</tr>
<tr>
<td>$\land$</td>
<td>2</td>
</tr>
<tr>
<td>$\lor$</td>
<td>3</td>
</tr>
<tr>
<td>$\rightarrow$</td>
<td>4</td>
</tr>
</tbody>
</table>
### Logic Computations Rules

<table>
<thead>
<tr>
<th>Equivalence</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p \lor F \equiv p$; $p \land T \equiv p$</td>
<td>Identity</td>
</tr>
<tr>
<td>$p \lor T \equiv T$; $p \land F \equiv F$</td>
<td>Domination</td>
</tr>
<tr>
<td>$p \lor p \equiv p$; $p \land p \equiv p$</td>
<td>Idempotent</td>
</tr>
<tr>
<td>$p \lor q \equiv q \lor p$; $p \land q \equiv q \land p$</td>
<td>Commutative</td>
</tr>
<tr>
<td>$p \lor (q \lor r) \equiv (p \lor q) \lor r$</td>
<td>Associative</td>
</tr>
<tr>
<td>$p \land (q \land r) \equiv (p \land (q \land r)$</td>
<td></td>
</tr>
<tr>
<td>$p \lor (q \land r) \equiv (p \lor q) \land (p \lor r)$</td>
<td>Distributive</td>
</tr>
<tr>
<td>$p \land (q \lor r) \equiv (p \lor q) \lor (p \land r)$</td>
<td></td>
</tr>
<tr>
<td>$\neg (p \land q) \equiv \neg p \lor \neg q$</td>
<td>De Morgan</td>
</tr>
<tr>
<td>$\neg (p \lor q) \equiv \neg p \land \neg q$</td>
<td></td>
</tr>
</tbody>
</table>

**Bảng:** Basic computation laws
summary

In this lecture we studied:

1. propositions

2. compound propositions

3. logical operators

4. truth tables

5. computation rules

6. we learned how to use truth tables to evaluate compound propositions

7. we conclude with two entertaining puzzles.
summary

In this lecture we studied:

1. propositions
2. compound propositions
In this lecture we studied:

1. propositions
2. compound propositions
3. logical operators
In this lecture we studied:

1. propositions
2. compound propositions
3. logical operators
4. truth tables
summary

In this lecture we studied:

1. propositions
2. compound propositions
3. logical operators
4. truth tables
5. computation rules

we learned how to use truth tables to evaluate compound propositions.

we conclude with two entertaining puzzles.
In this lecture we studied:

1. propositions
2. compound propositions
3. logical operators
4. truth tables
5. computation rules
6. we learned how to use truth tables to evaluate compound propositions

we conclude with two entertaining puzzles.
In this lecture we studied:

1. propositions
2. compound propositions
3. logical operators
4. truth tables
5. computation rules
6. we learned how to use truth tables to evaluate compound propositions
7. we conclude with two entertaining puzzles.
A simple puzzle

Trung, Hóa and Tuán had the same Pho for lunch at the same restaurant. They asked Hà to guess what they ate and where. To challenge Há they decided that each will tell Há two facts and at least one of the facts will be true.

- Trung: We ate Pho bò tái at Pho-24
- Hóa: We ate Pho gà at Quàn Ngon.
- Tuán: We ate at Cha Cá but definitely not Pho bò tái

What did they eat and where?
Trung, Hóa and Tuán had the same Pho for lunch at the same restaurant. They asked Hà to guess what they ate and where. To challenge Há they decided that each will tell Há two facts and at least one of the facts will be true.

- Trung: We ate Pho)bò tái at Pho-24
- Hóa: We ate Pho gà at Quàn ăn Ngon.
A simple puzzle

Trung, Hóa and Tuán had the same Pho for lunch at the same restaurant. They asked Hà to guess what they ate and where. To challenge Há they decided that each will tell Há two facts and at least one of the facts will be true.

- Trung: We ate Pho bò tái at Pho-24
- Hóa: We ate Pho gà at Quàn ăn Ngon.
- Tuán: We ate at Cha Cá but definitely not Pho bò tái
A simple puzzle

Trung, Hóa and Tuán had the same Pho for lunch at the same restaurant. They asked Hà to guess what they ate and where. To challenge Há they decided that each will tell Há two facts and at least one of the facts will be true..

- Trung: We ate Pho bò tái at Pho-24
- Hóa: We ate Pho gà at Quàn ăn Ngon.
- Tuán: We ate at Cha Cá but definitely not Pho bò tái
A simple puzzle

Trung, Hóa and Tuán had the same Pho for lunch at the same restaurant. They asked Hà to guess what they ate and where. To challenge Há they decided that each will tell Há two facts and at least one of the facts will be true.

- Trung: We ate Pho bò tái at Pho-24
- Hóa: We ate Pho gà at Quàn ăn Ngon.
- Tuán: We ate at Cha Cá but definitely not Pho bò tái

What did they eat and where?
To solve this puzzle we introduce five propositions:

- a: they ate Pho bò tái

The compound proposition describing their three claims is:

\[(a \lor b) \land (c \lor d) \land (\neg a \lor e) = \text{true}\]

which when expanded yields:

\[(a \land c \land \neg a) \lor (a \land c \land e) \lor (a \land d \land \neg a) \lor (a \land d \land e) \lor (b \land c \land \neg a) \lor (b \land c \land e) \lor (b \land d \land \neg a) \lor (b \land d \land e) = \text{true}\].

Among the eight triples there must be at least one which is true. For instance, \((b \land c \land e)\) would mean that they ate at two different restaurants which is false. The only triple which is true is \((b \land c \land \neg a)\) which says that they ate Pho gà at Pho-24.
To solve this puzzle we introduce five propositions:

- a: they ate Pho bò tái
- b: they ate at Pho-24

The compound proposition describing their three claims is:

\[(a \lor b) \land (c \lor d) \land (\neg a \lor e) = \text{true}\]

which when expanded yields:

\[(a \land c \land \neg a) \lor (a \land c \land e) \lor (a \land d \land \neg a) \lor (a \land d \land e) \lor (b \land c \land \neg a) \lor (b \land c \land e) \lor (b \land d \land \neg a) \lor (b \land d \land e) = \text{true}.

Among the eight triples there must be at least one which is true. For instance, \((b \land c \land e)\) would mean that they ate at two different restaurants which is false.

The only triple which is true is \((b \land c \land \neg a)\) which says that they ate Pho gà at Pho-24.
To solve this puzzle we introduce five propositions:

- a: they ate Pho bò tái
- b: they ate at Pho-24
- c: they ate Pho gà

The compound proposition describing their three claims is:

\[
(a \lor b) \land (c \lor d) \land (\neg a \lor e) = \text{true}
\]

which when expanded yields:

\[
(a \land c \land \neg a) \lor (a \land c \land e) \lor (a \land d \land \neg a) \lor (a \land d \land e) \lor (b \land c \land \neg a) \lor (b \land c \land e) \lor (b \land d \land \neg a) \lor (b \land d \land e) = \text{true}
\]
To solve this puzzle we introduce five propositions:

- a: they ate Pho bò tái
- b: they ate at Pho-24
- c: they ate Pho gà
- d: they ate at Quàn ăn Ngon

The compound proposition describing their three claims is:

\[(a \lor b) \land (c \lor d) \land (\neg a \lor e) = \text{true}\]

which when expanded yields:

\[(a \land c \land \neg a) \lor (a \land c \land e) \lor (a \land d \land \neg a) \lor (a \land d \land e) \lor (b \land c \land \neg a) \lor (b \land c \land e) \lor (b \land d \land \neg a) \lor (b \land d \land e) = \text{true}\].

Among the eight triples there must be at least one which is true. For instance, 

\[(b \land c \land e)\] would mean that they ate at two different restaurants which is false.

The only triple which is true is 

\[(b \land c \land \neg a)\] which says that they ate Pho gà at Pho-24.
To solve this puzzle we introduce five propositions:

- a: they ate Pho bò tái
- b: they ate at Pho-24
- c: they ate Pho gà
- d: they ate at Quân ăn Ngon
- e: they ate at Cha Cá

The compound proposition describing their three claims is:

\[(a \lor b) \land (c \lor d) \land (\neg a \lor e) = \text{true}\]

which when expanded yields:

\[(a \land c \land \neg a) \lor (a \land c \land e) \lor (a \land d \land \neg a) \lor (a \land d \land e) \lor (b \land c \land \neg a) \lor (b \land c \land e) \lor (b \land d \land \neg a) \lor (b \land d \land e) = \text{true}\].

Among the eight triples there must be at least one which is true. For instance, \((b \land c \land e)\) would mean that they ate at two different restaurants which is false.

The only triple which is true is \((b \land c \land \neg a)\) which says that they ate Pho gà at Pho-24.
To solve this puzzle we introduce five propositions:

- a: they ate Pho bò tái
- b: they ate at Pho-24
- c: they ate Pho gà
- d: they ate at Quàn ān Ngon
- e: they ate at Cha Cá

*The compound proposition describing their three claims is:*

\[(a \lor b) \land (c \lor d) \land (\neg a \lor e) = \text{true} \text{ which when expanded yields:} \]

\[(a \land c \land \neg a) \lor (a \land c \land e) \lor (a \land d \land \neg a) \lor (a \land d \land e) \lor (b \land c \land \neg a) \lor (b \land c \land e) \lor (b \land d \land \neg a) \lor (b \land d \land e) = \text{true}.\]
To solve this puzzle we introduce five propositions:

- a: they ate Phở bò tái
- b: they ate at Pho-24
- c: they ate Phở gà
- d: they ate at Quàn ĩn Ngon
- e: they ate at Cha Cá

The compound proposition describing their three claims is:

\[(a \lor b) \land (c \lor d) \land (\neg a \lor e) = \text{true}\] which when expanded yields:

\[(a \land c \land \neg a) \lor (a \land c \land e) \lor (a \land d \land \neg a) \lor (a \land d \land e) \lor (b \land c \land \neg a) \lor (b \land c \land e) \lor (b \land d \land \neg a) \lor (b \land d \land e) = \text{true}.\]
To solve this puzzle we introduce five propositions:

- a: they ate Pho bò tái
- b: they ate at Pho-24
- c: they ate Pho gà
- d: they ate at Quàn ăn Ngon
- e: they ate at Cha Cá

The compound proposition describing their three claims is:

\[(a \lor b) \land (c \lor d) \land (\neg a \lor e) = \text{true} \]

which when expanded yields:

\[
(a \land c \land \neg a) \lor (a \land c \land e) \lor (a \land d \land \neg a) \lor (a \land d \land e) \lor (b \land c \land \neg a) \lor (b \land c \land e) \lor (b \land d \land \neg a) \lor (b \land d \land e) = \text{true}.
\]

Among the eight triples there must be at least one which is true. For instance, \((b \land c \land e)\) would mean that they ate at two different restaurants which is false.
To solve this puzzle we introduce five propositions:

- a: they ate Pho bò tái
- b: they ate at Pho-24
- c: they ate Pho gà
- d: they ate at Quàn ăn Ngon
- e: they ate at Cha Cá

The compound proposition describing their three claims is:

\[(a \lor b) \land (c \lor d) \land (\neg a \lor e) = \text{true}\]

which when expanded yields:

\[
(a \land c \land \neg a) \lor (a \land c \land e) \lor (a \land d \land \neg a) \lor (a \land d \land e) \lor (b \land c \land \neg a) \lor (b \land c \land e) \lor (b \land d \land \neg a) \lor (b \land d \land e) = \text{true}.
\]

Among the eight triples there must be at least one which is true. For instance, \((b \land c \land e)\) would mean that they ate at two different restaurants which is false.

The only triple which is true is \((b \land c \land \neg a)\) which says that they ate Pho gà at Pho-24.
A logician is lost in a jungle in Africa. This jungle has two tribes of cannibals. One tribe always answer every question truthfully while the other tribe always lies.

A classic logic puzzle

Design a question that will guarantee to save the logician's life.
A logician is lost in a jungle in Africa. This jungle has two tribes of cannibals. One tribe always answers every question truthfully while the other tribe always lies.

The logician meets a cannibal in the jungle. The cannibal grants our logician his final wish: to ask him one question. If as a result of the answer the logician can determine to which tribe the cannibal belongs his life will be spared.
A classic logic puzzle

A logician is lost in a jungle in Africa. This jungle has two tribes of cannibals. One tribe always answer every question truthfully while the other tribe always lies.

The logician meets a cannibal in the jungle. The cannibal grants our logician his final wish: to ask him one question. If as a result of the answer the logician can determine to which tribe the cannibal belongs his life will be spared.

Design a question that will guarantee to save the logician’s life.