# Discrete Mathematics and its Applications

Ngày 14 tháng 9 năm 2011

# Logic-2

## (Applications)

*Applications of logic are abundent. Systems specifications where complex systems are designed use logic. Chips, web searches, programming languages, puzzle solving all use logic.*
*For instance, Sudoku puzzles can be solved using logic. Just introduce* 729 *propositions* $p(m, n, k)$ *which will mean "the number k goes in row m column n" then construct compound propositions that state that in every row, in every column and in the nine* $3 \times 3$ *blocks every integer from 1 to 9 must appear.*

# Logic-2

## (Applications)

*Applications of logic are abundent. Systems specifications where complex systems are designed use logic. Chips, web searches, programming languages, puzzle solving all use logic.*
*For instance, Sudoku puzzles can be solved using logic. Just introduce* 729 *propositions* $p(m, n, k)$ *which will mean "the number k goes in row m column n" then construct compound propositions that state that in every row, in every column and in the nine* $3 \times 3$ *blocks every integer from 1 to 9 must appear.*

# Logic-2

## (Applications)

*Applications of logic are abundent. Systems specifications where complex systems are designed use logic. Chips, web searches, programming languages, puzzle solving all use logic.*
*For instance, Sudoku puzzles can be solved using logic. Just introduce* 729 *propositions* $p(m,n,k)$ *which will mean "the number k goes in row m column n" then construct compound propositions that state that in every row, in every column and in the nine* $3 \times 3$ *blocks every integer from 1 to 9 must appear.*

*One of the most important development that had a profound impact on our life, is our ability to implement logical operations by electronic devices. The generic name of electronic devices that implement logic operations is* **logic gates**. *Imagine, withoout logic gates there would be no Facebook. Can you imagine living in a world without facebook?*

How important are logic gates? Walther Bothe received the Nobel Price in physics in 1954 for his development of the first electronic device that implemented the $\wedge$ gate.

How important are logic gates? Walther Bothe received the Nobel Price in physics in 1954 for his development of the first electronic device that implemented the $\wedge$ gate.

Today, we know how to implement logic gates at the molecular level.

How important are logic gates? Walther Bothe received the Nobel Price in physics in 1954 for his development of the first electronic device that implemented the $\wedge$ gate.

Today, we know how to implement logic gates at the molecular level.

Most programming languages implement four logical operators:

1. **not**

How important are logic gates? Walther Bothe received the Nobel Price in physics in 1954 for his development of the first electronic device that implemented the $\wedge$ gate.

Today, we know how to implement logic gates at the molecular level.

Most programming languages implement four logical operators:

1. **not**
2. **and**

How important are logic gates? Walther Bothe received the Nobel Price in physics in 1954 for his development of the first electronic device that implemented the $\wedge$ gate.

Today, we know how to implement logic gates at the molecular level.

Most programming languages implement four logical operators:

1. **not**
2. **and**
3. **or**

How important are logic gates? Walther Bothe received the Nobel Price in physics in 1954 for his development of the first electronic device that implemented the $\wedge$ gate.

Today, we know how to implement logic gates at the molecular level.

Most programming languages implement four logical operators:

1. **not**
2. **and**
3. **or**
4. **implies** (usually through the reserved word **if**)

How important are logic gates? Walther Bothe received the Nobel Price in physics in 1954 for his development of the first electronic device that implemented the $\wedge$ gate.

Today, we know how to implement logic gates at the molecular level.

Most programming languages implement four logical operators:

1. **not**
2. **and**
3. **or**
4. **implies** (usually through the reserved word **if**)

**Definition**

*Two propositions are* **logically equivalent** *if they have the same truth table.*

How important are logic gates? Walther Bothe received the Nobel Price in physics in 1954 for his development of the first electronic device that implemented the $\wedge$ gate.

Today, we know how to implement logic gates at the molecular level.

Most programming languages implement four logical operators:

1. **not**
2. **and**
3. **or**
4. **implies** (usually through the reserved word **if**)

**Definition**

*Two propositions are* **logically equivalent** *if they have the same truth table.*

**Example**

$p \vee (q \wedge r)$ *and* $(p \vee q) \wedge (p \vee r)$ *are logically equivalent.*

There are a few surprises when considering equivalence of propositions. Here is a list of some of them:

1. Are $(p \to q) \to s$ and $p \to (q \to s)$ equivalent?

There are a few surprises when considering equivalence of propositions. Here is a list of some of them:

1. Are $(p \to q) \to s$ and $p \to (q \to s)$ equivalent?
2. Are $p \mid q$ and $q \mid p$ equivalent?

There are a few surprises when considering equivalence of propositions. Here is a list of some of them:

1. Are $(p \to q) \to s$ and $p \to (q \to s)$ equivalent?
2. Are $p \mid q$ and $q \mid p$ equivalent?
3. What about $p \mid (q \mid s)$ and $(p \mid q) \mid s$?

There are a few surprises when considering equivalence of propositions. Here is a list of some of them:

1. Are $(p \to q) \to s$ and $p \to (q \to s)$ equivalent?
2. Are $p \mid q$ and $q \mid p$ equivalent?
3. What about $p \mid (q \mid s)$ and $(p \mid q) \mid s$?

### Definition

*A set of logical operators is a* **functionally complete collection** *if every truth table has an equivalent proposition using only operators from this set.*

There are a few surprises when considering equivalence of propositions. Here is a list of some of them:

1. Are $(p \to q) \to s$ and $p \to (q \to s)$ equivalent?
2. Are $p \mid q$ and $q \mid p$ equivalent?
3. What about $p \mid (q \mid s)$ and $(p \mid q) \mid s$?

### Definition

*A set of logical operators is a* **functionally complete collection** *if every truth table has an equivalent proposition using only operators from this set.*

### Comment

*In the next slide we shall see some examples of functionally complete collections. A bit surprising is that the $\mid$ and $\downarrow$ each by itself is a functionally complete collection. It is also noteworthy that the* **nand** *gate is the cheapest to build.*

The proposition $(p \vee q \vee r \vee s \vee t)$ is **false** only if the five propositions inside it are all **false**. In all other cases it will be **true**. Alternatively, its truth table has 32 entries, 31 of them are **true** and only one is **false**.

The proposition $(p \vee q \vee r \vee s \vee t)$ is **false** only if the five propositions inside it are all **false**. In all other cases it will be **true**. Alternatively, its truth table has 32 entries, 31 of them are **true** and only one is **false**.

Similary, the proposition $(p \wedge q \wedge \neg r \wedge s \wedge \neg t)$ is **true** if and only if the five propositions inside are all **true**. In all other cases it will be **false**. Alternatively, its truth table has 32 entries, 31 of them are **false** and onlyone is **true**.

Definition

The proposition ($p \vee q \vee r \vee s \vee t$) is **false** only if the five propositions inside it are all **false**. In all other cases it will be **true**. Alternatively, its truth table has 32 entries, 31 of them are **true** and only one is **false**.

Similary, the proposition ($p \wedge q \wedge \neg r \wedge s \wedge \neg t$) is **true** if and only if the five propositions inside are all **true**. In all other cases it will be **false**. Alternatively, its truth table has 32 entries, 31 of them are **false** and onlyone is **true**.

Definition

1. *A proposition of the form* ($p_1 \vee p_2 \vee \ldots \vee p_k$) *is called a* ***disjunction****.*

The proposition $(p \vee q \vee r \vee s \vee t)$ is **false** only if the five propositions inside it are all **false**. In all other cases it will be **true**. Alternatively, its truth table has 32 entries, 31 of them are **true** and only one is **false**.

Similary, the proposition $(p \wedge q \wedge \neg r \wedge s \wedge \neg t)$ is **true** if and only if the five propositions inside are all **true**. In all other cases it will be **false**. Alternatively, its truth table has 32 entries, 31 of them are **false** and onlyone is **true**.

### Definition

1. *A proposition of the form* $(p_1 \vee p_2 \vee \ldots \vee p_k)$ *is called a* **disjunction**.
2. *A proposition of the form* $(p_1 \wedge p_2 \wedge \ldots \wedge p_k)$ *is called a* **conjunction**.

The proposition ($p \vee q \vee r \vee s \vee t$) is **false** only if the five propositions inside it are all **false**. In all other cases it will be **true**. Alternatively, its truth table has 32 entries, 31 of them are **true** and only one is **false**.

Similary, the proposition ($p \wedge q \wedge \neg r \wedge s \wedge \neg t$) is **true** if and only if the five propositions inside are all **true**. In all other cases it will be **false**. Alternatively, its truth table has 32 entries, 31 of them are **false** and onlyone is **true**.

### Definition

1. *A proposition of the form* $(p_1 \vee p_2 \vee \ldots \vee p_k)$ *is called a* **disjunction**.
2. *A proposition of the form* $(p_1 \wedge p_2 \wedge \ldots \wedge p_k)$ *is called a* **conjunction**.
3. *A proposition of the form* $(D_1 \wedge D_2 \wedge \ldots D_m)$ *where each* $D_i$ *is a disjunction is called* **a conjunction of disjunctions**.

The proposition $(p \vee q \vee r \vee s \vee t)$ is **false** only if the five propositions inside it are all **false**. In all other cases it will be **true**. Alternatively, its truth table has 32 entries, 31 of them are **true** and only one is **false**.

Similary, the proposition $(p \wedge q \wedge \neg r \wedge s \wedge \neg t)$ is **true** if and only if the five propositions inside are all **true**. In all other cases it will be **false**. Alternatively, its truth table has 32 entries, 31 of them are **false** and onlyone is **true**.

Definition

1. *A proposition of the form $(p_1 \vee p_2 \vee \ldots \vee p_k)$ is called a* ***disjunction****.*
2. *A proposition of the form $(p_1 \wedge p_2 \wedge \ldots \wedge p_k)$ is called a* ***conjunction****.*
3. *A proposition of the form $(D_1 \wedge D_2 \wedge \ldots D_m)$ where each $D_i$ is a disjunction is called* **a conjunction of disjunctions***.*
4. *Similarly, a proposition of the form $(C_1 \vee C_2 \vee \ldots C_k)$ where each $C_i$ is a conjunction is called* **a disjunction of conjunctions***.*

## Theorem

*Every truth table represents a proposition which has an equivalent conjunction of disjunctions.*
*This implies that* $\neg, \vee, \wedge$ *is a functionally complete collection of logic operators.*

### Theorem

*Every truth table represents a proposition which has an equivalent conjunction of disjunctions.*
*This implies that* $\neg, \vee, \wedge$ *is a functionally complete collection of logic operators.*

## Theorem

*Every truth table represents a proposition which has an equivalent conjunction of disjunctions.*
*This implies that* $\neg, \vee, \wedge$ *is a functionally complete collection of logic operators.*

*Every truth table represents a proposition which has an equivalent disjunction of conjunctions.*

## Chứng minh.

Start writing... □

### Theorem

*Every truth table represents a proposition which has an equivalent conjunction of disjunctions.*
*This implies that* $\neg, \vee, \wedge$ *is a functionally complete collection of logic operators.*

*Every truth table represents a proposition which has an equivalent disjunction of conjunctions.*

### Chứng minh.

Start writing... $\square$

### Comment

*This explains why programming languages implement only four logical operators. All others can be easily implemented using them. See for example the Sage example. Even though as we shall soon see, we can use less operators, the choice to implement those is to make it more convenient for the programmer.*

**Definition**

*A proposition which is true for every assertion (assigning truth values to the variables) is called a* **tautology**.

## Definition

*A proposition which is true for every assertion (assigning truth values to the variables) is called a* **tautology**.

## Example

*The simplest example is the proposition* $p \vee \neg p$. *We will see other, more complicated examples in the exercises and drills.*

### Definition

*A proposition which is true for every assertion (assigning truth values to the variables) is called a* **tautology**.

### Example

*The simplest example is the proposition* $p \vee \neg p$. *We will see other, more complicated examples in the exercises and drills.*

### Definition

*A proposition which is false for all possible assertions is called a* **contradiction**.

### Definition

*A proposition which is true for every assertion (assigning truth values to the variables) is called a* **tautology**.

### Example

*The simplest example is the proposition $p \vee \neg p$. We will see other, more complicated examples in the exercises and drills.*

### Definition

*A proposition which is false for all possible assertions is called a* **contradiction**.

### Example

*Again, the simplest example is the proposition $p \wedge \neg p$.*