# HKOI Training

$ami \sim wkc$

Last modified: March 9, 2010

# Lecture 02

# Propositional Logic and Grammar in C

# First Order Propositions

## Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, “Yes” or “No”.[1]

# Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, “Yes” or “No”.[1]
For example, all the following are proposition.[2]

## Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, “Yes” or “No”.[1]
For example, all the following are proposition.[2]

- Today is hot.

# Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, “Yes” or “No”.[1]
For example, all the following are proposition.[2]

- Today is hot.
- I will not go to school.

# Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, "Yes" or "No".[1]
For example, all the following are proposition.[2]

- Today is hot.
- I will not go to school.
- $1+2+3=\frac{1}{2}(3)(4)$.

## Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, “Yes” or “No”.[1]
For example, all the following are proposition.[2]

- Today is hot.
- I will not go to school.
- $1 + 2 + 3 = \frac{1}{2}(3)(4)$. (Yes)

## Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, “Yes” or “No”.[1]
For example, all the following are proposition.[2]

- Today is hot.
- I will not go to school.
- $1 + 2 + 3 = \frac{1}{2}(3)(4)$. (Yes)
- There are infinitely many prime numbers.

## Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, “Yes” or “No”.[1]
For example, all the following are proposition.[2]

- Today is hot.
- I will not go to school.
- $1 + 2 + 3 = \frac{1}{2}(3)(4)$. (Yes)
- There are infinitely many prime numbers. (Yes)

## Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, “Yes” or “No”.[1] For example, all the following are proposition.[2]

- Today is hot.
- I will not go to school.
- $1 + 2 + 3 = \frac{1}{2}(3)(4)$. (Yes)
- There are infinitely many prime numbers. (Yes)
- $\sqrt{x^2} = x$.

## Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, “Yes” or “No”.[1]
For example, all the following are proposition.[2]

- Today is hot.
- I will not go to school.
- $1 + 2 + 3 = \frac{1}{2}(3)(4)$. (Yes)
- There are infinitely many prime numbers. (Yes)
- $\sqrt{x^2} = x$. (No, it is false when x is negative.)

## Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, "Yes" or "No".[1]
For example, all the following are proposition.[2]

- Today is hot.
- I will not go to school.
- $1 + 2 + 3 = \frac{1}{2}(3)(4)$. (Yes)
- There are infinitely many prime numbers. (Yes)
- $\sqrt{x^2} = x$. (No, it is false when x is negative.)
- If $n$ is a 5-digit square integer, then $n = 29929$.

# Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, “Yes” or “No”.[1]
For example, all the following are proposition.[2]

- Today is hot.
- I will not go to school.
- $1 + 2 + 3 = \frac{1}{2}(3)(4)$. (Yes)
- There are infinitely many prime numbers. (Yes)
- $\sqrt{x^2} = x$. (No, it is false when x is negative.)
- If $n$ is a 5-digit square integer, then $n = 29929$. (No)
- $x = 2$ only if $x^2 = 4$.

## Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, “Yes” or “No”.[1] For example, all the following are proposition.[2]

- Today is hot.
- I will not go to school.
- $1 + 2 + 3 = \frac{1}{2}(3)(4)$. (Yes)
- There are infinitely many prime numbers. (Yes)
- $\sqrt{x^2} = x$. (No, it is false when x is negative.)
- If $n$ is a 5-digit square integer, then $n = 29929$. (No)
- $x = 2$ only if $x^2 = 4$. (Yes)
- $x = 2$ if $x^2 = 4$.

# Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, “Yes” or “No”.[1]
For example, all the following are proposition.[2]

- Today is hot.
- I will not go to school.
- $1 + 2 + 3 = \frac{1}{2}(3)(4)$. (Yes)
- There are infinitely many prime numbers. (Yes)
- $\sqrt{x^2} = x$. (No, it is false when x is negative.)
- If $n$ is a 5-digit square integer, then $n = 29929$. (No)
- $x = 2$ only if $x^2 = 4$. (Yes)
- $x = 2$ if $x^2 = 4$. (No)
- $n = 2$ and $n$ is a prime.

## Proposition / Statement

A *statement* / *proposition* is a sentence that has either an answer, "Yes" or "No".[1] For example, all the following are proposition.[2]

- Today is hot.
- I will not go to school.
- $1 + 2 + 3 = \frac{1}{2}(3)(4)$. (Yes)
- There are infinitely many prime numbers. (Yes)
- $\sqrt{x^2} = x$. (No, it is false when x is negative.)
- If $n$ is a 5-digit square integer, then $n = 29929$. (No)
- $x = 2$ only if $x^2 = 4$. (Yes)
- $x = 2$ if $x^2 = 4$. (No)
- $n = 2$ and $n$ is a prime. (Yes)

[1] We skip a bit by using "common sense" to determine whether a sentence is a proposition or not.
[2] To emphasize that we are not solving equation, we interpret the $=$ sign to be "always equal".

# Proposition / Statement



The following are not propositions or we won't discuss the following kind of sentences.

## Proposition / Statement

The following are not propositions or we won't discuss the following kind of sentences.

- What time is it now?

## Proposition / Statement



The following are not propositions or we won't discuss the following kind of sentences.

- What time is it now?
-

## Proposition / Statement

The following are not propositions or we won't discuss the following kind of sentences.

- What time is it now?
- (empty string)[3]

## Proposition / Statement

The following are not propositions or we won't discuss the following kind of sentences.

- What time is it now?
- (empty string)[3]
- This statement is false.

# Proposition / Statement

The following are not propositions or we won't discuss the following kind of sentences.

- What time is it now?
- (empty string)[3]
- This statement is false.
- I am lying. [4]

## Proposition / Statement

The following are not propositions or we won't discuss the following kind of sentences.

- What time is it now?
- (empty string)[3]
- This statement is false.
- I am lying. [4]
- The second unique child of God is a female.

## Proposition / Statement

The following are not propositions or we won't discuss the following kind of sentences.

- What time is it now?
- (empty string)[3]
- This statement is false.
- I am lying. [4]
- The second unique child of God is a female.

Actually, some of them can be considered as statements.

However, for simplicity, we shall avoid them at this moment.

[3] This is usually called the $\epsilon$-string

[4] The Liar paradox

# Proposition operators

Given some propositions,

we can create new propositions from them by using *logical connectives*.

# Proposition operators



Given some propositions,

we can create new propositions from them by using *logical connectives*.

Be careful, we don't interpret the meaning at this stage.

# Proposition operators

Given some propositions,

we can create new propositions from them by using *logical connectives*.

Be careful, we don't interpret the meaning at this stage.

For example[5],

## Proposition operators

Given some propositions,

we can create new propositions from them by using *logical connectives*.

Be careful, we don’t interpret the meaning at this stage.

For example[5],

- NOT(Today is hot).

## Proposition operators

Given some propositions,

we can create new propositions from them by using *logical connectives*.

Be careful, we don't interpret the meaning at this stage.

For example[5],

- NOT(Today is hot).
- NOT(I will not go to school).

## Proposition operators

Given some propositions,

we can create new propositions from them by using *logical connectives*.

Be careful, we don't interpret the meaning at this stage.

For example[5],

- NOT(Today is hot).
- NOT(I will not go to school).
- Today is hot AND I will go to school.

## Proposition operators

Given some propositions,

we can create new propositions from them by using *logical connectives*.

Be careful, we don't interpret the meaning at this stage.

For example[5],

- NOT(Today is hot).
- NOT(I will not go to school).
- Today is hot AND I will go to school.
- If today is hot, then I will not go to school.

# Proposition operators

Given some propositions,

we can create new propositions from them by using *logical connectives*.

Be careful, we don't interpret the meaning at this stage.

For example[5],

- NOT(Today is hot).
- NOT(I will not go to school).
- Today is hot AND I will go to school.
- If today is hot, then I will not go to school.
- $x > 3$ OR $x < -1$.

## Proposition operators

Given some propositions,

we can create new propositions from them by using *logical connectives*.

Be careful, we don't interpret the meaning at this stage.

For example[5],

- NOT(Today is hot).
- NOT(I will not go to school).
- Today is hot AND I will go to school.
- If today is hot, then I will not go to school.
- $x > 3$ OR $x < -1$.
- Every $x$ is greater than $3$.

## Proposition operators

Given some propositions,

we can create new propositions from them by using *logical connectives*.

Be careful, we don't interpret the meaning at this stage.

For example[5],

- NOT(Today is hot).
- NOT(I will not go to school).
- Today is hot AND I will go to school.
- If today is hot, then I will not go to school.
- $x > 3$ OR $x < -1$.
- Every $x$ is greater than $3$.
- There is a number which is less than $-1$ or greater than $3$.

## Proposition operators

Given some propositions,

we can create new propositions from them by using *logical connectives*.

Be careful, we don't interpret the meaning at this stage.

For example[5],

- NOT(Today is hot).
- NOT(I will not go to school).
- Today is hot AND I will go to school.
- If today is hot, then I will not go to school.
- $x > 3$ OR $x < -1$.
- Every $x$ is greater than $3$.
- There is a number which is less than $-1$ or greater than $3$.

[5] We don't care about grammar or tense. What we are interested in the new proposition only.

## Negation - NOT

The negation of a proposition $P$ is $\sim P$.

# Negation - NOT

The negation of a proposition $P$ is $\sim P$.

Some book use $\neg P$ to denote the negation.

## Negation - NOT

The negation of a proposition $P$ is $\sim P$.

Some book use $\neg P$ to denote the negation.

It is simply a proposition prefixed by a word “not”.

- NOT(Today is hot).

## Negation - NOT

The negation of a proposition $P$ is $\sim P$.

Some book use $\neg P$ to denote the negation.

It is simply a proposition prefixed by a word “not”.

- NOT(Today is hot).
- NOT(I will not go to school).

## Negation - NOT

The negation of a proposition $P$ is $\sim P$.

Some book use $\neg P$ to denote the negation.

It is simply a proposition prefixed by a word “not”.

- NOT(Today is hot).
- NOT(I will not go to school).
- NOT($x > 3$).

# Negation - NOT

The negation of a proposition $P$ is $\sim P$.

Some book use $\neg P$ to denote the negation.

It is simply a proposition prefixed by a word “not”.

- NOT(Today is hot).
- NOT(I will not go to school).
- NOT($x > 3$).
- NOT($x$ is a prime).

## Negation - NOT

The negation of a proposition $P$ is $\sim P$.

Some book use $\neg P$ to denote the negation.

It is simply a proposition prefixed by a word “not”.

- NOT(Today is hot).
- NOT(I will not go to school).
- NOT($x > 3$).
- NOT($x$ is a prime).

# Conjunction - AND



The conjunction of two propositions $P, Q$ is $(P) \wedge (Q)$.

# Conjunction - AND

The conjunction of two propositions $P, Q$ is $(P) \wedge (Q)$.

We will denote the conjunction usually by $(P)\, and\, (Q)$ instead.

## Conjunction - AND

The conjunction of two propositions $P, Q$ is $(P) \wedge (Q)$.

We will denote the conjunction usually by $(P)\, and\, (Q)$ instead.

It connects two propositions by adding by a word “and”.

## Conjunction - AND



The conjunction of two propositions $P, Q$ is $(P) \wedge (Q)$.

We will denote the conjunction usually by $(P)\, and\, (Q)$ instead.

It connects two propositions by adding by a word “and”.

We may sometimes omit the parentheses as well as long as the meaning is clear.

# Conjunction - AND



The conjunction of two propositions $P, Q$ is $(P) \wedge (Q)$.

We will denote the conjunction usually by $(P)\, and\, (Q)$ instead.

It connects two propositions by adding by a word “and”.

We may sometimes omit the parentheses as well as long as the meaning is clear.

- (Today is hot) AND (I will go to school).

# Conjunction - AND



The conjunction of two propositions $P, Q$ is $(P) \wedge (Q)$.

We will denote the conjunction usually by $(P)\, and\, (Q)$ instead.

It connects two propositions by adding by a word “and”.

We may sometimes omit the parentheses as well as long as the meaning is clear.

- (Today is hot) AND (I will go to school).
- Today is hot AND I will go to school.

## Conjunction - AND

The conjunction of two propositions $P, Q$ is $(P) \wedge (Q)$.

We will denote the conjunction usually by $(P)\, and\, (Q)$ instead.

It connects two propositions by adding by a word “and”.

We may sometimes omit the parentheses as well as long as the meaning is clear.

- (Today is hot) AND (I will go to school).
- Today is hot AND I will go to school.
- NOT(I will not go to school) AND NOT(Today is hot).

# Conjunction - AND

The conjunction of two propositions $P, Q$ is $(P) \wedge (Q)$.

We will denote the conjunction usually by $(P)\, and\, (Q)$ instead.

It connects two propositions by adding by a word “and”.

We may sometimes omit the parentheses as well as long as the meaning is clear.

- (Today is hot) AND (I will go to school).
- Today is hot AND I will go to school.
- NOT(I will not go to school) AND NOT(Today is hot).
- ($x > 2$) AND ($x$ is even).

## Conjunction - AND



The conjunction of two propositions $P, Q$ is $(P) \wedge (Q)$.

We will denote the conjunction usually by $(P)\, and\, (Q)$ instead.

It connects two propositions by adding by a word “and”.

We may sometimes omit the parentheses as well as long as the meaning is clear.

- (Today is hot) AND (I will go to school).
- Today is hot AND I will go to school.
- NOT(I will not go to school) AND NOT(Today is hot).
- ($x > 2$) AND ($x$ is even).

# Disjunction - OR

The disjunction of two propositions $P, Q$ is $(P) \vee (Q)$.

# Disjunction - OR

The disjunction of two propositions $P, Q$ is $(P) \vee (Q)$.

We will denote the disjunction usually by $(P)\ or\ (Q)$ instead.

# Disjunction - OR

The disjunction of two propositions $P, Q$ is $(P) \vee (Q)$.

We will denote the disjunction usually by $(P)\ or\ (Q)$ instead.

It connects two propositions by adding by a word “or”.

## Disjunction - OR

The disjunction of two propositions $P, Q$ is $(P) \vee (Q)$.

We will denote the disjunction usually by $(P)\ or\ (Q)$ instead.

It connects two propositions by adding by a word “or”.

We may sometimes omit the parentheses as well as long as the meaning is clear.

# Disjunction - OR

The disjunction of two propositions $P, Q$ is $(P) \vee (Q)$.

We will denote the disjunction usually by $(P)\ or\ (Q)$ instead.

It connects two propositions by adding by a word “or”.

We may sometimes omit the parentheses as well as long as the meaning is clear.

- (Today is hot) OR (I will go to school).

# Disjunction - OR

The disjunction of two propositions $P, Q$ is $(P) \vee (Q)$.

We will denote the disjunction usually by $(P)\ or\ (Q)$ instead.

It connects two propositions by adding by a word “or”.

We may sometimes omit the parentheses as well as long as the meaning is clear.

- (Today is hot) OR (I will go to school).
- Today is hot OR I will go to school.

# Disjunction - OR

The disjunction of two propositions $P, Q$ is $(P) \vee (Q)$.

We will denote the disjunction usually by $(P)\ or\ (Q)$ instead.

It connects two propositions by adding by a word “or”.

We may sometimes omit the parentheses as well as long as the meaning is clear.

- (Today is hot) OR (I will go to school).
- Today is hot OR I will go to school.
- NOT(I will not go to school) OR NOT(Today is hot).

# Disjunction - OR

The disjunction of two propositions $P, Q$ is $(P) \vee (Q)$.

We will denote the disjunction usually by $(P)\ or\ (Q)$ instead.

It connects two propositions by adding by a word “or”.

We may sometimes omit the parentheses as well as long as the meaning is clear.

- (Today is hot) OR (I will go to school).
- Today is hot OR I will go to school.
- NOT(I will not go to school) OR NOT(Today is hot).

# Implication - IF-THEN



The implication of two propositions $P, Q$ is "IF $(P)$ THEN $(Q)$".

# Implication - IF-THEN

The implication of two propositions $P, Q$ is “IF $(P)$ THEN $(Q)$”.

We will denote the implication usually by “$P \implies Q$” instead.

## Implication - IF-THEN



The implication of two propositions $P, Q$ is “IF $(P)$ THEN $(Q)$”.

We will denote the implication usually by “$P \implies Q$” instead.

It connects two propositions by adding by an arrow or using the words “if” and “then”.

## Implication - IF-THEN



The implication of two propositions $P, Q$ is "IF $(P)$ THEN $(Q)$".

We will denote the implication usually by "$P \implies Q$" instead.

It connects two propositions by adding by an arrow or using the words "if" and "then".

We may sometimes omit the parentheses as well as long as the meaning is clear.

# Implication - IF-THEN

The implication of two propositions $P, Q$ is "IF $(P)$ THEN $(Q)$".

We will denote the implication usually by "$P \implies Q$" instead.

It connects two propositions by adding by an arrow or using the words "if" and "then".

We may sometimes omit the parentheses as well as long as the meaning is clear.

- IF (Today is hot) THEN (I will go to school).

# Implication - IF-THEN

The implication of two propositions $P, Q$ is "IF $(P)$ THEN $(Q)$".

We will denote the implication usually by "$P \implies Q$" instead.

It connects two propositions by adding by an arrow or using the words "if" and "then".

We may sometimes omit the parentheses as well as long as the meaning is clear.

- IF (Today is hot) THEN (I will go to school).
- IF (($x > 2$) AND ($x$ is even)) THEN (NOT($x$ is a prime)).

# Implication - IF-THEN

The implication of two propositions $P, Q$ is “IF $(P)$ THEN $(Q)$”.

We will denote the implication usually by “$P \implies Q$” instead.

It connects two propositions by adding by an arrow or using the words “if” and “then”.

We may sometimes omit the parentheses as well as long as the meaning is clear.

- IF (Today is hot) THEN (I will go to school).
- IF (($x > 2$) AND ($x$ is even)) THEN (NOT($x$ is a prime)).
- NOT(I will not go to school) OR NOT(Today is hot).

# Implication - IF-THEN

The implication of two propositions $P, Q$ is "IF $(P)$ THEN $(Q)$".

We will denote the implication usually by "$P \implies Q$" instead.

It connects two propositions by adding by an arrow or using the words "if" and "then".

We may sometimes omit the parentheses as well as long as the meaning is clear.

- IF (Today is hot) THEN (I will go to school).
- IF (($x > 2$) AND ($x$ is even)) THEN (NOT($x$ is a prime)).
- NOT(I will not go to school) OR NOT(Today is hot).

**Definition.** *Let $P$ and $Q$ be propositions,*

- *The* **converse** *of an implication $P \implies Q$ is $Q \implies P$.*[6]

## Implication - IF-THEN

The implication of two propositions $P, Q$ is “IF $(P)$ THEN $(Q)$”.
We will denote the implication usually by “$P \implies Q$” instead.
It connects two propositions by adding by an arrow or using the words “if” and “then”.
We may sometimes omit the parentheses as well as long as the meaning is clear.

- IF (Today is hot) THEN (I will go to school).
- IF (($x > 2$) AND ($x$ is even)) THEN (NOT($x$ is a prime)).
- NOT(I will not go to school) OR NOT(Today is hot).

**Definition.** *Let $P$ and $Q$ be propositions,*

- *The* **converse** *of an implication $P \implies Q$ is $Q \implies P$.*[6]
- *The* **inverse** *of an implication $P \implies Q$ is $\sim P \implies \sim Q$.*

## Implication - IF-THEN

The implication of two propositions $P, Q$ is "IF $(P)$ THEN $(Q)$".

We will denote the implication usually by "$P \implies Q$" instead.

It connects two propositions by adding by an arrow or using the words "if" and "then".

We may sometimes omit the parentheses as well as long as the meaning is clear.

- IF (Today is hot) THEN (I will go to school).
- IF (($x > 2$) AND ($x$ is even)) THEN (NOT($x$ is a prime)).
- NOT(I will not go to school) OR NOT(Today is hot).

**Definition.** *Let $P$ and $Q$ be propositions,*

- *The* **converse** *of an implication $P \implies Q$ is $Q \implies P$.*[6]
- *The* **inverse** *of an implication $P \implies Q$ is $\sim P \implies \sim Q$.*
- *The* **contrapositive** *of an implication $P \implies Q$ is $\sim Q \implies \sim P$.*

## Implication - IF-THEN

The implication of two propositions $P, Q$ is "IF $(P)$ THEN $(Q)$".

We will denote the implication usually by "$P \implies Q$" instead.

It connects two propositions by adding by an arrow or using the words "if" and "then".

We may sometimes omit the parentheses as well as long as the meaning is clear.

- IF (Today is hot) THEN (I will go to school).
- IF (($x > 2$) AND ($x$ is even)) THEN (NOT($x$ is a prime)).
- NOT(I will not go to school) OR NOT(Today is hot).

**Definition.** *Let $P$ and $Q$ be propositions,*

- *The* **converse** *of an implication $P \implies Q$ is $Q \implies P$.*[6]
- *The* **inverse** *of an implication $P \implies Q$ is $\sim P \implies \sim Q$.*
- *The* **contrapositive** *of an implication $P \implies Q$ is $\sim Q \implies \sim P$.*

[6] It is sometimes denoted by $P \impliedby Q$.

# Biconditional - IF-AND-ONLY-IF

The bi-conditional of two propositions $P, Q$ is "$(P)$ IF AND ONLY IF $(Q)$".

## Biconditional - IF-AND-ONLY-IF



The bi-conditional of two propositions $P, Q$ is "$(P)$ IF AND ONLY IF $(Q)$".

We will denote the bi-conditional usually by "$P \iff Q$" or "$P$ iff $Q$" instead.

## Biconditional - IF-AND-ONLY-IF



The bi-conditional of two propositions $P, Q$ is “$(P)$ IF AND ONLY IF $(Q)$”.

We will denote the bi-conditional usually by “$P \iff Q$” or “$P$ iff $Q$” instead.

It connects two propositions by adding by an bi-arrow.

## Biconditional - IF-AND-ONLY-IF



The bi-conditional of two propositions $P, Q$ is “$(P)$ IF AND ONLY IF $(Q)$”.

We will denote the bi-conditional usually by “$P \iff Q$” or “$P$ iff $Q$” instead.

It connects two propositions by adding by an bi-arrow.

- $(ax^2 + bx + c = 0$ has solution) IF AND ONLY IF $(b^2 - 4ac \geq 0)$ .

## Biconditional - IF-AND-ONLY-IF

The bi-conditional of two propositions $P, Q$ is “$(P)$ IF AND ONLY IF $(Q)$”.

We will denote the bi-conditional usually by “$P \iff Q$” or “$P$ iff $Q$” instead.

It connects two propositions by adding by an bi-arrow.

- ($ax^2 + bx + c = 0$ has solution) IF AND ONLY IF ($b^2 - 4ac \geq 0$) .
- ($n$ is a composite) IF AND ONLY IF (NOT($n$ is prime)).

## Biconditional - IF-AND-ONLY-IF

The bi-conditional of two propositions $P, Q$ is "$(P)$ IF AND ONLY IF $(Q)$".

We will denote the bi-conditional usually by "$P \iff Q$" or "$P$ iff $Q$" instead.

It connects two propositions by adding by an bi-arrow.

- ($ax^2 + bx + c = 0$ has solution) IF AND ONLY IF ($b^2 - 4ac \geq 0$) .
- ($n$ is a composite) IF AND ONLY IF (NOT($n$ is prime)).
- (Two lines are parallel) IF AND ONLY IF (NOT(They meet at a point)).

## Biconditional - IF-AND-ONLY-IF

The bi-conditional of two propositions $P, Q$ is "$(P)$ IF AND ONLY IF $(Q)$".

We will denote the bi-conditional usually by "$P \iff Q$" or "$P$ iff $Q$" instead.

It connects two propositions by adding by an bi-arrow.

- ($ax^2 + bx + c = 0$ has solution) IF AND ONLY IF ($b^2 - 4ac \geq 0$) .
- ($n$ is a composite) IF AND ONLY IF (NOT($n$ is prime)).
- (Two lines are parallel) IF AND ONLY IF (NOT(They meet at a point)).

We don't interpret the correctness of the above proposition, this is discussed in next section.

## Biconditional - IF-AND-ONLY-IF

The bi-conditional of two propositions $P, Q$ is "$(P)$ IF AND ONLY IF $(Q)$".

We will denote the bi-conditional usually by "$P \iff Q$" or "$P$ iff $Q$" instead.

It connects two propositions by adding by an bi-arrow.

- $(ax^2 + bx + c = 0$ has solution) IF AND ONLY IF ($b^2 - 4ac \geq 0$) .
- ($n$ is a composite) IF AND ONLY IF (NOT($n$ is prime)).
- (Two lines are parallel) IF AND ONLY IF (NOT(They meet at a point)).

We don't interpret the correctness of the above proposition, this is discussed in next section.

Indeed, if you consider the correctness, not all of them are always true.

# Boolean value

# Truth value



Truth value are a value, either "*false*" or "*true*", associated to each proposition.

## Truth value

Truth value are a value, either "*false*" or "*true*", associated to each proposition.

The association of the value must obey the following laws:

# Truth value

Truth value are a value, either "*false*" or "*true*", associated to each proposition.

The association of the value must obey the following laws:

1. Each proposition has ONE value each time.

2. Every propositions constructed by propositional operators will have a corresponding value.

## Truth value

Truth value are a value, either "*false*" or "*true*", associated to each proposition.
The association of the value must obey the following laws:

1. Each proposition has ONE value each time.
2. Every propositions constructed by propositional operators will have a corresponding value.

**Definition.** *A proposition that is always having the truth value "true" is called a* tautology*.*
*A proposition that is always having the truth value "false" is called a* contradiction*.*

# Truth value

Truth value are a value, either "*false*" or "*true*", associated to each proposition.
The association of the value must obey the following laws:

1. Each proposition has ONE value each time.
2. Every propositions constructed by propositional operators will have a corresponding value.

**Definition.** *A proposition that is always having the truth value "true" is called a* tautology*.*
*A proposition that is always having the truth value "false" is called a* contradiction*.*

To demonstrate the first law,
for example, using our common sense to associate the truth value to the following:

- $1 + 1$ is equal to $2$.

# Truth value

Truth value are a value, either "*false*" or "*true*", associated to each proposition.
The association of the value must obey the following laws:

1. Each proposition has ONE value each time.
2. Every propositions constructed by propositional operators will have a corresponding value.

**Definition.** *A proposition that is always having the truth value "true" is called a* tautology*.*
*A proposition that is always having the truth value "false" is called a* contradiction*.*

To demonstrate the first law,
for example, using our common sense to associate the truth value to the following:

- $1 + 1$ is equal to $2$. (Tautology)

## Truth value

Truth value are a value, either "*false*" or "*true*", associated to each proposition.
The association of the value must obey the following laws:

1. Each proposition has ONE value each time.

2. Every propositions constructed by propositional operators will have a corresponding value.

**Definition.** *A proposition that is always having the truth value "true" is called a* tautology.
*A proposition that is always having the truth value "false" is called a* contradiction.

To demonstrate the first law,
for example, using our common sense to associate the truth value to the following:

- $1 + 1$ is equal to $2$. (Tautology)

- The Earth is a square.

# Truth value

Truth value are a value, either "*false*" or "*true*", associated to each proposition.
The association of the value must obey the following laws:

1. Each proposition has ONE value each time.
2. Every propositions constructed by propositional operators will have a corresponding value.

**Definition.** *A proposition that is always having the truth value "true" is called a* tautology*.*
*A proposition that is always having the truth value "false" is called a* contradiction*.*

To demonstrate the first law,
for example, using our common sense to associate the truth value to the following:

- $1+1$ is equal to $2$. (Tautology)
- The Earth is a square. (Contradiction)

# Truth value

Truth value are a value, either "*false*" or "*true*", associated to each proposition.
The association of the value must obey the following laws:

1. Each proposition has ONE value each time.
2. Every propositions constructed by propositional operators will have a corresponding value.

**Definition.** *A proposition that is always having the truth value "true" is called a* tautology. *A proposition that is always having the truth value "false" is called a* contradiction.

To demonstrate the first law,
for example, using our common sense to associate the truth value to the following:

- $1 + 1$ is equal to $2$. (Tautology)
- The Earth is a square. (Contradiction)
- Today is hot.

## Truth value

Truth value are a value, either "*false*" or "*true*", associated to each proposition.
The association of the value must obey the following laws:

1. Each proposition has ONE value each time.
2. Every propositions constructed by propositional operators will have a corresponding value.

**Definition.** *A proposition that is always having the truth value "true" is called a* tautology*.*
*A proposition that is always having the truth value "false" is called a* contradiction*.*

To demonstrate the first law,
for example, using our common sense to associate the truth value to the following:

- $1 + 1$ is equal to $2$. (Tautology)
- The Earth is a square. (Contradiction)
- Today is hot. (Just a proposition)

# Truth value

Truth value are a value, either "*false*" or "*true*", associated to each proposition.
The association of the value must obey the following laws:

1. Each proposition has ONE value each time.
2. Every propositions constructed by propositional operators will have a corresponding value.

**Definition.** *A proposition that is always having the truth value "true" is called a* tautology.
*A proposition that is always having the truth value "false" is called a* contradiction.

To demonstrate the first law,
for example, using our common sense to associate the truth value to the following:

- $1 + 1$ is equal to $2$. (Tautology)
- The Earth is a square. (Contradiction)
- Today is hot. (Just a proposition)

## Boolean operations - NOT

Suppose $P$ is a proposition and it has a truth value.

## Boolean operations - NOT

Suppose $P$ is a proposition and it has a truth value.

Are the truth value of $P$ and $\sim P$ related?

## Boolean operations - NOT

Suppose $P$ is a proposition and it has a truth value.

Are the truth value of $P$ and $\sim P$ related?

The second law state that they are related according to some rules, which is given as follow.

# Boolean operations - NOT

Suppose $P$ is a proposition and it has a truth value.

Are the truth value of $P$ and $\sim P$ related?

The second law state that they are related according to some rules, which is given as follow.

| $P$ | $\sim P$ |
|---|---|
| true | false |
| false | true |

## Boolean operations - NOT

Suppose $P$ is a proposition and it has a truth value.

Are the truth value of $P$ and $\sim P$ related?

The second law state that they are related according to some rules, which is given as follow.

| $P$ | $\sim P$ |
|---|---|
| true | false |
| false | true |

That means, whenever $P$ is associated with a value “true” , $\sim P$ must have the value “false”.

## Boolean operations - NOT

Suppose $P$ is a proposition and it has a truth value.

Are the truth value of $P$ and $\sim P$ related?

The second law state that they are related according to some rules, which is given as follow.

| $P$ | $\sim P$ |
|---|---|
| true | false |
| false | true |

That means, whenever $P$ is associated with a value “true” , $\sim P$ must have the value “false”.

And whenever $P$ is associated with a value “false” , $\sim P$ must have the value “true”.

# Boolean operations - NOT



Suppose $P$ is a proposition and it has a truth value.

Are the truth value of $P$ and $\sim P$ related?

The second law state that they are related according to some rules, which is given as follow.

| $P$ | $\sim P$ |
|---|---|
| true | false |
| false | true |

That means, whenever $P$ is associated with a value “true” , $\sim P$ must have the value “false”.

And whenever $P$ is associated with a value “false” , $\sim P$ must have the value “true”.

# Boolean operations - AND



Suppose $P$ and $Q$ are propositions and have truth value.

# Boolean operations - AND

Suppose $P$ and $Q$ are propositions and have truth value.

Similarly, the truth value of “$P$ and $Q$” are related by the following table.

# Boolean operations - AND

Suppose $P$ and $Q$ are propositions and have truth value.

Similarly, the truth value of “$P$ and $Q$” are related by the following table.

| $P$ | $Q$ | $P$ and $Q$ |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | false |

## Boolean operations - AND

Suppose $P$ and $Q$ are propositions and have truth value.

Similarly, the truth value of “$P$ and $Q$” are related by the following table.

| $P$ | $Q$ | $P$ and $Q$ |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | false |

To interpret the table, it is equal to ask whether both propositions are true.

# Boolean operations - OR

Suppose $P$ and $Q$ are propositions and have truth value.

## Boolean operations - OR

Suppose $P$ and $Q$ are propositions and have truth value.

Similarly, the truth value of "$P$ or $Q$" are related by the following table.

# Boolean operations - OR



Suppose $P$ and $Q$ are propositions and have truth value.

Similarly, the truth value of "$P$ or $Q$" are related by the following table.

| $P$ | $Q$ | $P$ or $Q$ |
|---|---|---|
| true | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

## Boolean operations - OR

Suppose $P$ and $Q$ are propositions and have truth value.

Similarly, the truth value of “$P$ or $Q$” are related by the following table.

| $P$ | $Q$ | $P$ or $Q$ |
|---|---|---|
| true | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

To interpret the table, it is equal to ask whether at least one of the propositions is true.

## Boolean operations - IF-THEN



Suppose $P$ and $Q$ are propositions and have truth value.

# Boolean operations - IF-THEN



Suppose $P$ and $Q$ are propositions and have truth value.

Similarly, the truth value of "$P \implies Q$" are related by the following table.

# Boolean operations - IF-THEN

Suppose $P$ and $Q$ are propositions and have truth value.

Similarly, the truth value of "$P \implies Q$" are related by the following table.

| $P$ | $Q$ | $P \implies Q$ |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | true |
| false | false | true |

## Boolean operations - IF-THEN



Suppose $P$ and $Q$ are propositions and have truth value.

Similarly, the truth value of "$P \implies Q$" are related by the following table.

| $P$ | $Q$ | $P \implies Q$ |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | true |
| false | false | true |

It is difficult at first to accept this table.

## Boolean operations - IF-THEN



Suppose $P$ and $Q$ are propositions and have truth value.

Similarly, the truth value of "$P \implies Q$" are related by the following table.

| $P$ | $Q$ | $P \implies Q$ |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | true |
| false | false | true |

It is difficult at first to accept this table.

For example,

the proposition "IF (The earth is a square) THEN $(1+1=3)$" has a value "true".

## Boolean operations - IF-THEN

Suppose $P$ and $Q$ are propositions and have truth value.

Similarly, the truth value of "$P \implies Q$" are related by the following table.

| $P$ | $Q$ | $P \implies Q$ |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | true |
| false | false | true |

It is difficult at first to accept this table.

For example,

the proposition "IF (The earth is a square) THEN $(1 + 1 = 3)$" has a value "true".

The correct interpretation is that

"whether one can determine the statement is honest or not and if so, is it honest?"

# Boolean operations - IF-THEN



Suppose $P$ and $Q$ are propositions and have truth value.

Similarly, the truth value of "$P \implies Q$" are related by the following table.

| $P$ | $Q$ | $P \implies Q$ |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | true |
| false | false | true |

It is difficult at first to accept this table.

For example,

the proposition "IF (The earth is a square) THEN $(1+1=3)$" has a value "true".

The correct interpretation is that

"whether one can determine the statement is honest or not and if so, is it honest?"

One can determine a people is lying only when the condition holds,

otherwise we can say that is a joke rather than a lie.

## Boolean operations - IF-AND-ONLY-IF



Suppose $P$ and $Q$ are propositions and have truth value.

# Boolean operations - IF-AND-ONLY-IF

Suppose $P$ and $Q$ are propositions and have truth value.

The truth value of “$P \iff Q$” are related by the following table.

## Boolean operations - IF-AND-ONLY-IF

Suppose $P$ and $Q$ are propositions and have truth value.

The truth value of "$P \iff Q$" are related by the following table.

| $P$ | $Q$ | $P \iff Q$ |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | true |

## Boolean operations - IF-AND-ONLY-IF

Suppose $P$ and $Q$ are propositions and have truth value.

The truth value of "$P \iff Q$" are related by the following table.

| $P$ | $Q$ | $P \iff Q$ |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | true |

The bi-condition is true only when both propositions have the same truth value.

## Boolean operations - IF-AND-ONLY-IF

Suppose $P$ and $Q$ are propositions and have truth value.

The truth value of "$P \iff Q$" are related by the following table.

| $P$ | $Q$ | $P \iff Q$ |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | true |

The bi-condition is true only when both propositions have the same truth value.

**Definition.** *Let $P$ and $Q$ be two propositions,*
*$P$ and $Q$ are* **logically equivalent** *if $P \iff Q$.*

# Boolean operations - IF-AND-ONLY-IF

Suppose $P$ and $Q$ are propositions and have truth value.

The truth value of "$P \iff Q$" are related by the following table.

| $P$ | $Q$ | $P \iff Q$ |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | true |

The bi-condition is true only when both propositions have the same truth value.

**Definition.** *Let $P$ and $Q$ be two propositions, $P$ and $Q$ are* **logically equivalent** *if $P \iff Q$.*

The equivalence is in a sense that

by merely looking at the truth value of two propositions, we cannot distinguish them.

## Boolean operations - IF-AND-ONLY-IF

Suppose $P$ and $Q$ are propositions and have truth value.

The truth value of "$P \iff Q$" are related by the following table.

| $P$ | $Q$ | $P \iff Q$ |
|---|---|---|
| true | true | true |
| true | false | false |
| false | true | false |
| false | false | true |

The bi-condition is true only when both propositions have the same truth value.

**Definition.** *Let $P$ and $Q$ be two propositions,*
*$P$ and $Q$ are* **logically equivalent** *if $P \iff Q$.*

The equivalence is in a sense that

by merely looking at the truth value of two propositions, we cannot distinguish them.

So, that means the two propositions are logically the same.

**Theorem.** *Let $P_1$ and $P_2$ be two propositions,*
*and $P_1$ :="$P \iff Q$", $P_2$ :="$(P \implies Q)$ and $(Q \implies Q)$".*
*$P_1$ and $P_2$ are logically equivalent.*

## Boolean Algebra

Let $P$, $Q$ and $R$ be propositions, $\mathcal{T}$ be a tautology and $\mathcal{F}$ be a contradiction.

Prove that the following pairs are equivalent:

| | |
|---|---|
| $\sim \mathcal{T}$ | $\mathcal{F}$ |
| $\sim \mathcal{F}$ | $\mathcal{T}$ |
| $\sim\ \sim P$ | $P$ |
| $P$ and $\sim P$ | $\mathcal{F}$ |
| $P$ or $\sim P$ | $\mathcal{T}$ |
| $\sim (P$ and $Q)$ | $\sim P$ or $\sim Q$ |
| $\sim (P$ or $Q)$ | $\sim P$ and $\sim Q$ |
| ($P$ and $Q$) and ($R$) | ($P$) and ($Q$ and $R$) |
| ($P$ or $Q$) or ($R$) | ($P$) or ($Q$ or $R$) |
| ($P$ and $Q$) or ($R$) | ($P$ or $R$) and ($Q$ or $R$) |
| ($P$ or $Q$) and ($R$) | ($P$ and $R$) or ($Q$ and $R$) |
| ($P$ or $Q$) and ($P$) | $P$ |
| ($P$ and $Q$) or ($P$) | $P$ |

## Example

Let $Q$ and $R$ be propositions,

# Example

Let $Q$ and $R$ be propositions,

$P_1$ be the proposition that "$Q \implies R$",

# Example

Let $Q$ and $R$ be propositions,
$P_1$ be the proposition that "$Q \implies R$",
$P_2$ be the proposition that "not $Q$ or $R$".

# Example

Let $Q$ and $R$ be propositions,
$P_1$ be the proposition that "$Q \implies R$",
$P_2$ be the proposition that "not $Q$ or $R$".
$P_3$ be the proposition that "(not $R$) $\implies$ (not $Q$)"

# Example

Let $Q$ and $R$ be propositions,
$P_1$ be the proposition that "$Q \implies R$",
$P_2$ be the proposition that "not $Q$ or $R$".
$P_3$ be the proposition that "(not $R$) $\implies$ (not $Q$)"

*Proof.* We first show that $P_1 \iff P_2$ is true by computing all cases.

| $Q$ | $R$ | $\sim Q$ | $Q \implies R$ | $\sim Q$ or $R$ | $P_1 \iff P_2$ |
|---|---|---|---|---|---|

# Example

Let $Q$ and $R$ be propositions,
$P_1$ be the proposition that "$Q \implies R$",
$P_2$ be the proposition that "not $Q$ or $R$".
$P_3$ be the proposition that "(not $R$) $\implies$ (not $Q$)"

*Proof.* We first show that $P_1 \iff P_2$ is true by computing all cases.

| $Q$ | $R$ | $\sim Q$ | $Q \implies R$ | $\sim Q$ or $R$ | $P_1 \iff P_2$ |
|---|---|---|---|---|---|
| true | true | false | true | true | true |

## Example

Let $Q$ and $R$ be propositions,
$P_1$ be the proposition that "$Q \implies R$",
$P_2$ be the proposition that "not $Q$ or $R$".
$P_3$ be the proposition that "(not $R$) $\implies$ (not $Q$)"

*Proof.* We first show that $P_1 \iff P_2$ is true by computing all cases.

| $Q$ | $R$ | $\sim Q$ | $Q \implies R$ | $\sim Q$ or $R$ | $P_1 \iff P_2$ |
|---|---|---|---|---|---|
| true | true | false | true | true | true |
| true | false | false | false | false | true |

# Example

Let $Q$ and $R$ be propositions,

$P_1$ be the proposition that "$Q \implies R$",

$P_2$ be the proposition that "not $Q$ or $R$".

$P_3$ be the proposition that "(not $R$) $\implies$ (not $Q$)"

*Proof.* We first show that $P_1 \iff P_2$ is true by computing all cases.

| $Q$ | $R$ | $\sim Q$ | $Q \implies R$ | $\sim Q$ or $R$ | $P_1 \iff P_2$ |
|---|---|---|---|---|---|
| true | true | false | true | true | true |
| true | false | false | false | false | true |
| false | true | true | true | true | true |

## Example

Let $Q$ and $R$ be propositions,

$P_1$ be the proposition that "$Q \implies R$",

$P_2$ be the proposition that "not $Q$ or $R$".

$P_3$ be the proposition that "(not $R$) $\implies$ (not $Q$)"

*Proof.* We first show that $P_1 \iff P_2$ is true by computing all cases.

| $Q$ | $R$ | $\sim Q$ | $Q \implies R$ | $\sim Q$ or $R$ | $P_1 \iff P_2$ |
|---|---|---|---|---|---|
| true | true | false | true | true | true |
| true | false | false | false | false | true |
| false | true | true | true | true | true |
| false | false | true | true | true | true |

# Example

Let $Q$ and $R$ be propositions,
$P_1$ be the proposition that “$Q \implies R$”,
$P_2$ be the proposition that “not $Q$ or $R$”.
$P_3$ be the proposition that “(not $R$) $\implies$ (not $Q$)”

*Proof.* We first show that $P_1 \iff P_2$ is true by computing all cases.

| $Q$ | $R$ | $\sim Q$ | $Q \implies R$ | $\sim Q$ or $R$ | $P_1 \iff P_2$ |
|---|---|---|---|---|---|
| true | true | false | true | true | true |
| true | false | false | false | false | true |
| false | true | true | true | true | true |
| false | false | true | true | true | true |

## Example

Let $Q$ and $R$ be propositions,
$P_1$ be the proposition that "$Q \implies R$",
$P_2$ be the proposition that "not $Q$ or $R$".
$P_3$ be the proposition that "(not $R$) $\implies$ (not $Q$)"

*Proof.* We first show that $P_1 \iff P_2$ is true by computing all cases.

| $Q$ | $R$ | $\sim Q$ | $Q \implies R$ | $\sim Q$ or $R$ | $P_1 \iff P_2$ |
|---|---|---|---|---|---|
| true | true | false | true | true | true |
| true | false | false | false | false | true |
| false | true | true | true | true | true |
| false | false | true | true | true | true |

Next, we show that $P_2 \iff P_3$ as follow:
$P_3 = \sim R \implies \sim Q \iff \sim(\sim R)$ or $\sim Q$
the proposition $\sim(\sim R)$ or$Q \iff R$ or $\sim Q = P_2$ $\square$

## Example

Let $P$ be the proposition that "$n$ is a five-digit square integer whose digits are all $2$ and $9$",
$Q$ be the proposition that "$n$ is $29929$."
The above two are equivalent.

## Example

Let $P$ be the proposition that "$n$ is a five-digit square integer whose digits are all $2$ and $9$", $Q$ be the proposition that "$n$ is $29929$."

The above two are equivalent.

*Proof.* Show that $P \implies Q$ and $Q \implies P$.

$Q \implies P$: Check that $29929 = 173^2$.

$P \implies Q$: Read lecture 1. $\square$

# Second Order Propositions

# Propositional variables

A proposition may be depend on variable(s).

## Propositional variables

A proposition may be depend on variable(s).

For example, we let $P(n)$ be the proposition that “$n$ is a prime number.”.

## Propositional variables

A proposition may be depend on variable(s).

For example, we let $P(n)$ be the proposition that "$n$ is a prime number.".

Then we have infinitely many propositions depends on $n$, say

# Propositional variables



A proposition may be depend on variable(s).

For example, we let $P(n)$ be the proposition that "$n$ is a prime number.".

Then we have infinitely many propositions depends on $n$, say

- $P(6)$ is the proposition "$6$ is a prime number" .

## Propositional variables

A proposition may be depend on variable(s).

For example, we let $P(n)$ be the proposition that “$n$ is a prime number.”.

Then we have infinitely many propositions depends on $n$, say

- $P(6)$ is the proposition “$6$ is a prime number” .
- $P(11)$ is the proposition “$11$ is a prime number” .

# Propositional variables

A proposition may be depend on variable(s).

For example, we let $P(n)$ be the proposition that “$n$ is a prime number.”.

Then we have infinitely many propositions depends on $n$, say

- $P(6)$ is the proposition “$6$ is a prime number” .
- $P(11)$ is the proposition “$11$ is a prime number” .
- $P(123)$ is the proposition “$123$ is a prime number” .

# Propositional variables

A proposition may be depend on variable(s).

For example, we let $P(n)$ be the proposition that "$n$ is a prime number.".

Then we have infinitely many propositions depends on $n$, say

- $P(6)$ is the proposition "$6$ is a prime number" .
- $P(11)$ is the proposition "$11$ is a prime number" .
- $P(123)$ is the proposition "$123$ is a prime number" .
- ...

## Propositional variables

A proposition may be depend on variable(s).

For example, we let $P(n)$ be the proposition that "$n$ is a prime number.".

Then we have infinitely many propositions depends on $n$, say

- $P(6)$ is the proposition "$6$ is a prime number" .
- $P(11)$ is the proposition "$11$ is a prime number" .
- $P(123)$ is the proposition "$123$ is a prime number" .
- ...

Let $Q(x, y)$ be the proposition that "$x$ is smaller than $y$" [7]

## Propositional variables

A proposition may be depend on variable(s).

For example, we let $P(n)$ be the proposition that "$n$ is a prime number.".

Then we have infinitely many propositions depends on $n$, say

- $P(6)$ is the proposition "$6$ is a prime number" .
- $P(11)$ is the proposition "$11$ is a prime number" .
- $P(123)$ is the proposition "$123$ is a prime number" .
- ...

Let $Q(x, y)$ be the proposition that "$x$ is smaller than $y$" [7]

For example, $Q(\text{John, Mary})$ is the proposition that "John is smaller than Mary".

## Propositional variables

A proposition may be depend on variable(s).

For example, we let $P(n)$ be the proposition that "$n$ is a prime number.".

Then we have infinitely many propositions depends on $n$, say

- $P(6)$ is the proposition "$6$ is a prime number" .
- $P(11)$ is the proposition "$11$ is a prime number" .
- $P(123)$ is the proposition "$123$ is a prime number" .
- ...

Let $Q(x, y)$ be the proposition that "$x$ is smaller than $y$" [7]

For example, $Q(\text{John}, \text{Mary})$ is the proposition that "John is smaller than Mary".

$Q(2, 3)$ is the proposition that "2 is smaller than 3".

## Propositional variables

A proposition may be depend on variable(s).

For example, we let $P(n)$ be the proposition that "$n$ is a prime number.".

Then we have infinitely many propositions depends on $n$, say

- $P(6)$ is the proposition "$6$ is a prime number" .
- $P(11)$ is the proposition "$11$ is a prime number" .
- $P(123)$ is the proposition "$123$ is a prime number" .
- ...

Let $Q(x, y)$ be the proposition that "$x$ is smaller than $y$" [7]

For example, $Q(\text{John, Mary})$ is the proposition that "John is smaller than Mary".

$Q(2, 3)$ is the proposition that "2 is smaller than 3".

[7] The values of a variable need not be a number.

# Existential quantifier - THERE EXISTS

As like what we did for those first order logical connectives,

## Existential quantifier - THERE EXISTS



As like what we did for those first order logical connectives,

we can construct new proposition by using second order logical connectives.

## Existential quantifier - THERE EXISTS

As like what we did for those first order logical connectives,

we can construct new proposition by using second order logical connectives.

Let $P(n)$ be the proposition that "$n$ is a prime number.".

# Existential quantifier - THERE EXISTS

As like what we did for those first order logical connectives,

we can construct new proposition by using second order logical connectives.

Let $P(n)$ be the proposition that "$n$ is a prime number.".

We can create a new proposition that "There is an integer $k$ such that $P(k)$".

## Existential quantifier - THERE EXISTS

As like what we did for those first order logical connectives,
we can construct new proposition by using second order logical connectives.
Let $P(n)$ be the proposition that “$n$ is a prime number.”.
We can create a new proposition that “There is an integer $k$ such that $P(k)$”.
It is denoted by “$\exists k(P(k))$”.

# Existential quantifier - THERE EXISTS

As like what we did for those first order logical connectives,

we can construct new proposition by using second order logical connectives.

Let $P(n)$ be the proposition that "$n$ is a prime number.".

We can create a new proposition that "There is an integer $k$ such that $P(k)$".

It is denoted by "$\exists k(P(k))$".

However, to avoid so many parentheses, it is usually denoted as "$\exists k,\ P(k)$".

Its truth values depends on all the proposition $P(n)$,

# Existential quantifier - THERE EXISTS

As like what we did for those first order logical connectives,

we can construct new proposition by using second order logical connectives.

Let $P(n)$ be the proposition that “$n$ is a prime number.”.

We can create a new proposition that “There is an integer $k$ such that $P(k)$”.

It is denoted by “$\exists k(P(k))$”.

However, to avoid so many parentheses, it is usually denoted as “$\exists k,\ P(k)$”.

Its truth values depends on all the proposition $P(n)$,

it is true if there is at least one proposition having the value “true”.

## Existential quantifier - THERE EXISTS

As like what we did for those first order logical connectives,

we can construct new proposition by using second order logical connectives.

Let $P(n)$ be the proposition that "$n$ is a prime number.".

We can create a new proposition that "There is an integer $k$ such that $P(k)$".

It is denoted by "$\exists k(P(k))$".

However, to avoid so many parentheses, it is usually denoted as "$\exists k,\ P(k)$".

Its truth values depends on all the proposition $P(n)$,

it is true if there is at least one proposition having the value "true".

Since $P(2)$ is true, "$\exists k,\ P(k)$" is true.

# Existential quantifier - THERE EXISTS



As like what we did for those first order logical connectives,

we can construct new proposition by using second order logical connectives.

Let $P(n)$ be the proposition that "$n$ is a prime number.".

We can create a new proposition that "There is an integer $k$ such that $P(k)$".

It is denoted by "$\exists k(P(k))$".

However, to avoid so many parentheses, it is usually denoted as "$\exists k,\ P(k)$".

Its truth values depends on all the proposition $P(n)$,

it is true if there is at least one proposition having the value "true".

Since $P(2)$ is true, "$\exists k,\ P(k)$" is true.

Simply because there is such an integer.

## Universial quantifier - FOR ALL

Let $P(n)$ be the proposition that “$n$ is a prime number.”.

## Universial quantifier - FOR ALL



Let $P(n)$ be the proposition that "$n$ is a prime number.".

We can create a new proposition that "Every integer $k$ such that $P(k)$".

# Universal quantifier - FOR ALL

Let $P(n)$ be the proposition that “$n$ is a prime number.”.

We can create a new proposition that “Every integer $k$ such that $P(k)$”.

It is denoted by “$\forall k(P(k))$”.

# Universial quantifier - FOR ALL

Let $P(n)$ be the proposition that “$n$ is a prime number.”.

We can create a new proposition that “Every integer $k$ such that $P(k)$”.

It is denoted by “$\forall k(P(k))$”.

However, to avoid so many parentheses, it is usually denoted as “$\forall k,\ P(k)$”.

Its truth values depends on all the proposition $P(n)$,

# Universal quantifier - FOR ALL

Let $P(n)$ be the proposition that "$n$ is a prime number.".

We can create a new proposition that "Every integer $k$ such that $P(k)$".

It is denoted by "$\forall k(P(k))$".

However, to avoid so many parentheses, it is usually denoted as "$\forall k,\ P(k)$".

Its truth values depends on all the proposition $P(n)$,

it is true if all propositions are having the value "true".

## Universial quantifier - FOR ALL



Let $P(n)$ be the proposition that "$n$ is a prime number.".

We can create a new proposition that "Every integer $k$ such that $P(k)$".

It is denoted by "$\forall k(P(k))$".

However, to avoid so many parentheses, it is usually denoted as "$\forall k,\ P(k)$".

Its truth values depends on all the proposition $P(n)$,

it is true if all propositions are having the value "true".

As $P(4)$ is false, "$\forall k,\ P(k)$" is false.

## Universial quantifier - FOR ALL

Let $P(n)$ be the proposition that "$n$ is a prime number.".

We can create a new proposition that "Every integer $k$ such that $P(k)$".

It is denoted by "$\forall k(P(k))$".

However, to avoid so many parentheses, it is usually denoted as "$\forall k,\ P(k)$".

Its truth values depends on all the proposition $P(n)$,

it is true if all propositions are having the value "true".

As $P(4)$ is false, "$\forall k,\ P(k)$" is false.

Simply because not all of them are "true".

**Theorem.** *The following are true:*

*"not(*$\forall x, P(x)$*)* $\iff$ $\exists x$*, not*$P(x)$*"*[8]

*"not(*$\exists x, P(x)$*)* $\iff$ $\forall x$*, not*$P(x)$*"*[9]

**Theorem.** *The following are true:*

*"not(*$\forall x, P(x)$*)* $\iff$ $\exists x$, *not*$P(x)$*"*[8]

*"not(*$\exists x, P(x)$*)* $\iff$ $\forall x$, *not*$P(x)$*"*[9]

Question:

**Theorem.** *The following are true:*

*“not($\forall x, P(x)$) $\iff$ $\exists x$, not$P(x)$”*[8]

*“not($\exists x, P(x)$) $\iff$ $\forall x$, not$P(x)$”*[9]

Question:

Can we interchange the operators FOR ALL and THERE EXISTS?

i.e. Are the two proposition “$\forall x, \exists y, Q(x, y)$” , “$\exists y, \forall x, Q(x, y)$” the same?

**Theorem.** *The following are true:*

*"not($\forall x, P(x)$) $\iff$ $\exists x$, not$P(x)$"*[8]

*"not($\exists x, P(x)$) $\iff$ $\forall x$, not$P(x)$"*[9]

Question:

Can we interchange the operators FOR ALL and THERE EXISTS?

i.e. Are the two proposition "$\forall x, \exists y, Q(x, y)$" , "$\exists y, \forall x, Q(x, y)$" the same?

Hints: Let $Q(x, y)$ be the proposition that "$x$ is smaller than $y$".

**Theorem.** *The following are true:*

*"not($\forall x, P(x)$) $\iff$ $\exists x$, not$P(x)$"*[8]

*"not($\exists x, P(x)$) $\iff$ $\forall x$, not$P(x)$"*[9]

Question:

Can we interchange the operators FOR ALL and THERE EXISTS?

i.e. Are the two proposition "$\forall x, \exists y, Q(x, y)$" , "$\exists y, \forall x, Q(x, y)$" the same?

Hints: Let $Q(x, y)$ be the proposition that "$x$ is smaller than $y$".

"$\forall x, \exists y, Q(x, y)$" means

for every number $x$, there is an number $y$ such that $x < y$.

**Theorem.** *The following are true:*

*"not($\forall x, P(x)$) $\iff$ $\exists x$, not$P(x)$"*[8]

*"not($\exists x, P(x)$) $\iff$ $\forall x$, not$P(x)$"*[9]

Question:

Can we interchange the operators FOR ALL and THERE EXISTS?

i.e. Are the two proposition "$\forall x, \exists y, Q(x, y)$" , "$\exists y, \forall x, Q(x, y)$" the same?

Hints: Let $Q(x, y)$ be the proposition that "$x$ is smaller than $y$".

"$\forall x, \exists y, Q(x, y)$" means

for every number $x$, there is an number $y$ such that $x < y$.

i.e. For each given number, there is a larger number.

**Theorem.** *The following are true:*

*"not($\forall x, P(x)$) $\iff$ $\exists x$, not$P(x)$"*[8]

*"not($\exists x, P(x)$) $\iff$ $\forall x$, not$P(x)$"*[9]

Question:

Can we interchange the operators FOR ALL and THERE EXISTS?

i.e. Are the two proposition "$\forall x, \exists y, Q(x, y)$" , "$\exists y, \forall x, Q(x, y)$" the same?

Hints: Let $Q(x, y)$ be the proposition that "$x$ is smaller than $y$".

"$\forall x, \exists y, Q(x, y)$" means

for every number $x$, there is an number $y$ such that $x < y$.

i.e. For each given number, there is a larger number.

"$\exists y, \forall x, Q(x, y)$" means

there is an number $y$ such that every number $x$ is smaller than $y$.

[8] Not every proposition are true means that at least one of them is false.

[9] Not having at least one true means that all of them are false.

# Recursive definition

## Quick introduction

a **recursive definition** (or **inductive definition**) is used to define an object in terms of itself[10].

For example,

let $a_n$ be numbers defined as follows:

1. $a_0 = 1$
2. $a_n = n \cdot a_{n-1}$ , for $n > 0$

To find $a_6$, we put $n = 6$ and use the second one, $a_6 = 6 \cdot a_5$ and so on...

## Quick introduction

a **recursive definition** (or **inductive definition**) is used to define an object in terms of itself[10].

For example,

let $a_n$ be numbers defined as follows:

1. $a_0 = 1$
2. $a_n = n \cdot a_{n-1}$ , for $n > 0$

To find $a_6$, we put $n = 6$ and use the second one, $a_6 = 6 \cdot a_5$ and so on...

until we arrive at $a_0$ which is a known value and we therefore know $a_6 = 720$.

## Quick introduction

a **recursive definition** (or **inductive definition**) is used to define an object in terms of itself[10].
For example,
let $a_n$ be numbers defined as follows:

1. $a_0 = 1$
2. $a_n = n \cdot a_{n-1}$ , for $n > 0$

To find $a_6$, we put $n = 6$ and use the second one, $a_6 = 6 \cdot a_5$ and so on...
until we arrive at $a_0$ which is a known value and we therefore know $a_6 = 720$.

**Definition.** *The $n$-**th factorial**, $n!$, is defined as the value of $a_n$ as above.*

## Quick introduction

a **recursive definition** (or **inductive definition**) is used to define an object in terms of itself[10].
For example,
let $a_n$ be numbers defined as follows:

1. $a_0 = 1$
2. $a_n = n \cdot a_{n-1}$ , for $n > 0$

To find $a_6$, we put $n = 6$ and use the second one, $a_6 = 6 \cdot a_5$ and so on...
until we arrive at $a_0$ which is a known value and we therefore know $a_6 = 720$.

**Definition.** *The $n$-**th factorial**, $n!$, is defined as the value of $a_n$ as above.*

Recursive definition is something like above as long as the objects involved are well-defined.

## Quick introduction

a **recursive definition** (or **inductive definition**) is used to define an object in terms of itself[10].
For example,
let $a_n$ be numbers defined as follows:

1. $a_0 = 1$
2. $a_n = n \cdot a_{n-1}$ , for $n > 0$

To find $a_6$, we put $n = 6$ and use the second one, $a_6 = 6 \cdot a_5$ and so on...
until we arrive at $a_0$ which is a known value and we therefore know $a_6 = 720$.

**Definition.** *The $n$-**th factorial**, $n!$, is defined as the value of $a_n$ as above.*

Recursive definition is something like above as long as the objects involved are well-defined.

[10] P. Aczel (1977), "An introduction to inductive definitions", Handbook of Mathematical Logic, J. Barwise (ed.)

# Polynomial evaluation

Let $f$ be a polynomial and $f(x) = 3x^5 + 2x^3 + 5x - 7$.

## Polynomial evaluation

Let $f$ be a polynomial and $f(x) = 3x^5 + 2x^3 + 5x - 7$.
To find $f(2)$, we usually compute $3 \cdot 2^5, 2 \cdot x^3, \cdots$ and add them together.

# Polynomial evaluation



Let $f$ be a polynomial and $f(x) = 3x^5 + 2x^3 + 5x - 7$.

To find $f(2)$, we usually compute $3 \cdot 2^5, 2 \cdot x^3, \cdots$ and add them together.

Here is a better method for find the value $f(2)$.

## Polynomial evaluation

Let $f$ be a polynomial and $f(x) = 3x^5 + 2x^3 + 5x - 7$.
To find $f(2)$, we usually compute $3 \cdot 2^5, 2 \cdot x^3, \cdots$ and add them together.
Here is a better method for find the value $f(2)$.

Define $a_n$ be the coefficient of $x^n$ of $f(x)$,
i.e. $a_0 = -7, a_1 = 5, a_2 = 0, a_3 = 2, a_4 = 0, a_5 = 3$
let $x_n$ be numbers defined as follows:

1. $x_0 = a_5$
2. $x_n = 2 \cdot x_{n-1} + a_{5-n}$ , for $5 \geq n > 0$

The value of $f(2)$ is exactly $x_5$.

# Polynomial evaluation

Let $f$ be a polynomial and $f(x) = 3x^5 + 2x^3 + 5x - 7$.

To find $f(2)$, we usually compute $3 \cdot 2^5, 2 \cdot x^3, \cdots$ and add them together.

Here is a better method for find the value $f(2)$.

Define $a_n$ be the coefficient of $x^n$ of $f(x)$,

i.e. $a_0 = -7, a_1 = 5, a_2 = 0, a_3 = 2, a_4 = 0, a_5 = 3$

let $x_n$ be numbers defined as follows:

1. $x_0 = a_5$
2. $x_n = 2 \cdot x_{n-1} + a_{5-n}$ , for $5 \geq n > 0$

The value of $f(2)$ is exactly $x_5$.

What are the other numbers $x_0, \cdots, x_4$ used for?

## Polynomial evaluation

Let $f$ be a polynomial and $f(x) = 3x^5 + 2x^3 + 5x - 7$.
To find $f(2)$, we usually compute $3 \cdot 2^5, 2 \cdot x^3, \cdots$ and add them together.
Here is a better method for find the value $f(2)$.

Define $a_n$ be the coefficient of $x^n$ of $f(x)$,
i.e. $a_0 = -7, a_1 = 5, a_2 = 0, a_3 = 2, a_4 = 0, a_5 = 3$
let $x_n$ be numbers defined as follows:

1. $x_0 = a_5$
2. $x_n = 2 \cdot x_{n-1} + a_{5-n}$ , for $5 \geq n > 0$

The value of $f(2)$ is exactly $x_5$.
What are the other numbers $x_0, \cdots, x_4$ used for?
Try to divide $f(x)$ by $(x - 2)$ using polynomial long division.

## Polynomial evaluation

Let $f$ be a polynomial and $f(x) = 3x^5 + 2x^3 + 5x - 7$.
To find $f(2)$, we usually compute $3 \cdot 2^5, 2 \cdot x^3, \cdots$ and add them together.
Here is a better method for find the value $f(2)$.

Define $a_n$ be the coefficient of $x^n$ of $f(x)$,
i.e. $a_0 = -7, a_1 = 5, a_2 = 0, a_3 = 2, a_4 = 0, a_5 = 3$
let $x_n$ be numbers defined as follows:

1. $x_0 = a_5$
2. $x_n = 2 \cdot x_{n-1} + a_{5-n}$ , for $5 \geq n > 0$

The value of $f(2)$ is exactly $x_5$.
What are the other numbers $x_0, \cdots, x_4$ used for?
Try to divide $f(x)$ by $(x-2)$ using polynomial long division.
If we let $b_n = x_{4-n}$ for $0 \leq n \leq 4$, then the polynomial
$Q(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4$ is exactly the quotient.
Recursive definition is something like about as long as the objects involved are well-defined.

# Polynomial evaluation

Let $f$ be a polynomial and $f(x) = 3x^5 + 2x^3 + 5x - 7$.
To find $f(2)$, we usually compute $3 \cdot 2^5, 2 \cdot x^3, \cdots$ and add them together.
Here is a better method for find the value $f(2)$.

Define $a_n$ be the coefficient of $x^n$ of $f(x)$,
i.e. $a_0 = -7, a_1 = 5, a_2 = 0, a_3 = 2, a_4 = 0, a_5 = 3$
let $x_n$ be numbers defined as follows:

1. $x_0 = a_5$
2. $x_n = 2 \cdot x_{n-1} + a_{5-n}$ , for $5 \geq n > 0$

The value of $f(2)$ is exactly $x_5$.
What are the other numbers $x_0, \cdots, x_4$ used for?
Try to divide $f(x)$ by $(x-2)$ using polynomial long division.
If we let $b_n = x_{4-n}$ for $0 \leq n \leq 4$, then the polynomial
$Q(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4$ is exactly the quotient.
Recursive definition is something like about as long as the objects involved are well-defined.

## Polynomial evaluation

$$
\begin{array}{r|rrrrrr}
 & 3x^4 & +6x^3 & +14x^2 & +28x & +61 & \\
\cline{2-7}
x-2 & 3x^5 & & +2x^3 & & +5x & -7 \\
 & -3x^5+6x^4 & & & & & \\
\cline{2-2}
 & & 6x^4 & +2x^3 & & & \\
 & & -6x^4 & +12x^3 & & & \\
\cline{3-4}
 & & & 14x^3 & & & \\
 & & & -14x^3 & +28x^2 & & \\
\cline{4-5}
 & & & & 28x^2 & +5x & \\
 & & & & -28x^2 & +56x & \\
\cline{5-6}
 & & & & & 61x & -7 \\
 & & & & & -61x & +122 \\
\cline{6-7}
 & & & & & & 115
\end{array}
$$

# Synthetic substitution



Read the Extra Material

# Grammar in C

# English Grammar

For those who study linguistic, they view the grammar of English systematically.

- Each passage consists of a title, author and a sequence of paragraphs

# English Grammar

For those who study linguistic, they view the grammar of English systematically.

- Each passage consists of a title, author and a sequence of paragraphs
- Each paragraph is a sequence of sentences

# English Grammar



For those who study linguistic, they view the grammar of English systematically.

- Each passage consists of a title, author and a sequence of paragraphs
- Each paragraph is a sequence of sentences
- Every sentence end with a full-stop (.) or an exclamation mark (!).

# English Grammar

For those who study linguistic, they view the grammar of English systematically.

- Each passage consists of a title, author and a sequence of paragraphs
- Each paragraph is a sequence of sentences
- Every sentence end with a full-stop (.) or an exclamation mark (!).
- <sentence> ::= <subject> <verb> [<object>] [11]

# English Grammar

For those who study linguistic, they view the grammar of English systematically.

- Each passage consists of a title, author and a sequence of paragraphs
- Each paragraph is a sequence of sentences
- Every sentence end with a full-stop (.) or an exclamation mark (!).
- <sentence> ::= <subject> <verb> [<object>] [11]
- Every subject is a noun-phrase, verb-phrase etc.

## English Grammar

For those who study linguistic, they view the grammar of English systematically.

- Each passage consists of a title, author and a sequence of paragraphs
- Each paragraph is a sequence of sentences
- Every sentence end with a full-stop (.) or an exclamation mark (!).
- <sentence> ::= <subject> <verb> [<object>] [11]
- Every subject is a noun-phrase, verb-phrase etc.
- Every noun-pharse are consists of a smaller noun-phrase, noun, relative pronoun etc.

# English Grammar



For those who study linguistic, they view the grammar of English systematically.

- Each passage consists of a title, author and a sequence of paragraphs
- Each paragraph is a sequence of sentences
- Every sentence end with a full-stop (.) or an exclamation mark (!).
- <sentence> ::= <subject> <verb> [<object>] [11]
- Every subject is a noun-phrase, verb-phrase etc.
- Every noun-pharse are consists of a smaller noun-phrase, noun, relative pronoun etc.
- Every noun are vocabulary.

# English Grammar

For those who study linguistic, they view the grammar of English systematically.

- Each passage consists of a title, author and a sequence of paragraphs
- Each paragraph is a sequence of sentences
- Every sentence end with a full-stop (.) or an exclamation mark (!).
- <sentence> ::= <subject> <verb> [<object>] [11]
- Every subject is a noun-phrase, verb-phrase etc.
- Every noun-pharse are consists of a smaller noun-phrase, noun, relative pronoun etc.
- Every noun are vocabulary.
- Every vocabulary is consisting of correctly spelled character sequences.

# English Grammar

For those who study linguistic, they view the grammar of English systematically.

- Each passage consists of a title, author and a sequence of paragraphs
- Each paragraph is a sequence of sentences
- Every sentence end with a full-stop (.) or an exclamation mark (!).
- <sentence> ::= <subject> <verb> [<object>] [11]
- Every subject is a noun-phrase, verb-phrase etc.
- Every noun-pharse are consists of a smaller noun-phrase, noun, relative pronoun etc.
- Every noun are vocabulary.
- Every vocabulary is consisting of correctly spelled character sequences.
- A character sequence is consists of characters

# English Grammar

For those who study linguistic, they view the grammar of English systematically.

- Each passage consists of a title, author and a sequence of paragraphs
- Each paragraph is a sequence of sentences
- Every sentence end with a full-stop (.) or an exclamation mark (!).
- <sentence> ::= <subject> <verb> [<object>] [11]
- Every subject is a noun-phrase, verb-phrase etc.
- Every noun-pharse are consists of a smaller noun-phrase, noun, relative pronoun etc.
- Every noun are vocabulary.
- Every vocabulary is consisting of correctly spelled character sequences.
- A character sequence is consists of characters
- Each character is from ’a’ to ’z’ or ’A’ to ’Z’

# English Grammar

For those who study linguistic, they view the grammar of English systematically.

- Each passage consists of a title, author and a sequence of paragraphs
- Each paragraph is a sequence of sentences
- Every sentence end with a full-stop (.) or an exclamation mark (!).
- <sentence> ::= <subject> <verb> [<object>] [11]
- Every subject is a noun-phrase, verb-phrase etc.
- Every noun-pharse are consists of a smaller noun-phrase, noun, relative pronoun etc.
- Every noun are vocabulary.
- Every vocabulary is consisting of correctly spelled character sequences.
- A character sequence is consists of characters
- Each character is from ’a’ to ’z’ or ’A’ to ’Z’

[11] Every sentence consists of a subject and a verb, object is optional

## C Grammar - BNF Grammar

An analogue linguistic structure for programming language C also exists.

- Each program consists of pre-processor directives and functions[12]

# C Grammar - BNF Grammar

An analogue linguistic structure for programming language C also exists.

- Each program consists of pre-processor directives and functions[12]
- Each function consists of statements.

# C Grammar - BNF Grammar

An analogue linguistic structure for programming language C also exists.

- Each program consists of pre-processor directives and functions[12]
- Each function consists of statements.
- Every statements ends with a semi-colon (;).

## C Grammar - BNF Grammar

An analogue linguistic structure for programming language C also exists.

- Each program consists of pre-processor directives and functions[12]
- Each function consists of statements.
- Every statements ends with a semi-colon (;).
- Every statement is either a control flow , variable declaration or expression.

# C Grammar - BNF Grammar

An analogue linguistic structure for programming language C also exists.

- Each program consists of pre-processor directives and functions[12]
- Each function consists of statements.
- Every statements ends with a semi-colon (;).
- Every statement is either a control flow , variable declaration or expression.
- expression can be arithmetic expression, logical expression etc.

# C Grammar - BNF Grammar

An analogue linguistic structure for programming language C also exists.

- Each program consists of pre-processor directives and functions[12]
- Each function consists of statements.
- Every statements ends with a semi-colon (;).
- Every statement is either a control flow , variable declaration or expression.
- expression can be arithmetic expression, logical expression etc.
- arithmetic expression consists consists of addition, subtraction etc.

## C Grammar - BNF Grammar

An analogue linguistic structure for programming language C also exists.

- Each program consists of pre-processor directives and functions[12]
- Each function consists of statements.
- Every statements ends with a semi-colon (;).
- Every statement is either a control flow , variable declaration or expression.
- expression can be arithmetic expression, logical expression etc.
- arithmetic expression consists consists of addition, subtraction etc.
- <addition>::=<Arithmetic Operand 1> ’+’ <Arithmetic Operand 2>

# C Grammar - BNF Grammar

An analogue linguistic structure for programming language C also exists.

- Each program consists of pre-processor directives and functions[12]
- Each function consists of statements.
- Every statements ends with a semi-colon (;).
- Every statement is either a control flow , variable declaration or expression.
- expression can be arithmetic expression, logical expression etc.
- arithmetic expression consists consists of addition, subtraction etc.
- <addition>::=<Arithmetic Operand 1> '+' <Arithmetic Operand 2>
- Arithmetic Operand can be function values, variables names or numbers.

# C Grammar - BNF Grammar

An analogue linguistic structure for programming language C also exists.

- Each program consists of pre-processor directives and functions[12]
- Each function consists of statements.
- Every statements ends with a semi-colon (;).
- Every statement is either a control flow , variable declaration or expression.
- expression can be arithmetic expression, logical expression etc.
- arithmetic expression consists consists of addition, subtraction etc.
- <addition>::=<Arithmetic Operand 1> '+' <Arithmetic Operand 2>
- Arithmetic Operand can be function values, variables names or numbers.
- numbers can be integers or real numbers.

## C Grammar - BNF Grammar



An analogue linguistic structure for programming language C also exists.

- Each program consists of pre-processor directives and functions[12]
- Each function consists of statements.
- Every statements ends with a semi-colon (;).
- Every statement is either a control flow , variable declaration or expression.
- expression can be arithmetic expression, logical expression etc.
- arithmetic expression consists consists of addition, subtraction etc.
- <addition>::=<Arithmetic Operand 1> '+' <Arithmetic Operand 2>
- Arithmetic Operand can be function values, variables names or numbers.
- numbers can be integers or real numbers.
- <integers>::= [0-9]+, i.e. at least one of any one of '0','1', $\cdots$ , '9'

## C Grammar - BNF Grammar

An analogue linguistic structure for programming language C also exists.

- Each program consists of pre-processor directives and functions[12]
- Each function consists of statements.
- Every statements ends with a semi-colon (;).
- Every statement is either a control flow , variable declaration or expression.
- expression can be arithmetic expression, logical expression etc.
- arithmetic expression consists consists of addition, subtraction etc.
- <addition>::=<Arithmetic Operand 1> '+' <Arithmetic Operand 2>
- Arithmetic Operand can be function values, variables names or numbers.
- numbers can be integers or real numbers.
- <integers>::= [0-9]+, i.e. at least one of any one of '0','1', $\cdots$ , '9'

[12] A part to tell the computer to store action.

# Arithmetic Expression



We learnt the way to convert usual mathematical expression into C language.

## Arithmetic Expression



We learnt the way to convert usual mathematical expression into C language.

However, how could we tell the computer to do the following two different action?

# Arithmetic Expression



We learnt the way to convert usual mathematical expression into C language.

However, how could we tell the computer to do the following two different action?

$3 \div 2$ Quotient: 1

$3 \div 2$

## Arithmetic Expression

We learnt the way to convert usual mathematical expression into C language.

However, how could we tell the computer to do the following two different action?

| | |
|---|---|
| $3 \div 2$ | Quotient: 1 |
| $3 \div 2$ | Real division: 1.5 |

## Arithmetic Expression

We learnt the way to convert usual mathematical expression into C language.

However, how could we tell the computer to do the following two different action?

| | |
|---|---|
| $3 \div 2$ | Quotient: 1 |
| $3 \div 2$ | Real division: 1.5 |

# Arithmetic Expression



We learnt the way to convert usual mathematical expression into C language.

However, how could we tell the computer to do the following two different action?

$3 \div 2$ Quotient: 1

$3 \div 2$ Real division: 1.5

Even ourselves cannot distinguish the two different division without further explanation.

# Arithmetic Expression

We learnt the way to convert usual mathematical expression into C language.

However, how could we tell the computer to do the following two different action?

| | |
|---|---|
| $3 \div 2$ | Quotient: 1 |
| $3 \div 2$ | Real division: 1.5 |

Even ourselves cannot distinguish the two different division without further explanation.

Computer will use the following rules to distinguish the two different division.

# Arithmetic Expression

We learnt the way to convert usual mathematical expression into C language.

However, how could we tell the computer to do the following two different action?

| | |
|---|---|
| $3 \div 2$ | Quotient: 1 |
| $3 \div 2$ | Real division: 1.5 |

Even ourselves cannot distinguish the two different division without further explanation.

Computer will use the following rules to distinguish the two different division.

1. If every operands are computer-integers, it perform the quotient division.[13]

# Arithmetic Expression

We learnt the way to convert usual mathematical expression into C language.

However, how could we tell the computer to do the following two different action?

$3 \div 2$ Quotient: 1

$3 \div 2$ Real division: 1.5

Even ourselves cannot distinguish the two different division without further explanation.

Computer will use the following rules to distinguish the two different division.

1. If every operands are computer-integers, it perform the quotient division.[13]
2. If any one of the operand is not a computer-integer, it will switch to real division.

## Arithmetic Expression



We learnt the way to convert usual mathematical expression into C language.

However, how could we tell the computer to do the following two different action?

| | |
|---|---|
| $3 \div 2$ | Quotient: 1 |
| $3 \div 2$ | Real division: 1.5 |

Even ourselves cannot distinguish the two different division without further explanation.

Computer will use the following rules to distinguish the two different division.

1. If every operands are computer-integers, it perform the quotient division.[13]
2. If any one of the operand is not a computer-integer, it will switch to real division.

Computer-integers means that the number is “written in a form”
so that the computer treat it as an integer.

# Arithmetic Expression

We learnt the way to convert usual mathematical expression into C language.

However, how could we tell the computer to do the following two different action?

| | |
|---|---|
| $3 \div 2$ | Quotient: 1 |
| $3 \div 2$ | Real division: 1.5 |

Even ourselves cannot distinguish the two different division without further explanation.

Computer will use the following rules to distinguish the two different division.

1. If every operands are computer-integers, it perform the quotient division.[13]
2. If any one of the operand is not a computer-integer, it will switch to real division.

Computer-integers means that the number is "written in a form"
so that the computer treat it as an integer.
For example, we can say $2.3$ is not a whole number due to the decimal place.

[13] Computer like doing discrete mathematics too!

# Regular Expression



To specify clear what we mean by “written in a form”, we need **Regular Expression**.

# Regular Expression

To specify clear what we mean by “written in a form”, we need **Regular Expression**.

It is a tool used for string matching.

# Regular Expression

To specify clear what we mean by “written in a form”, we need **Regular Expression**.

It is a tool used for string matching.

For example,

# Regular Expression



To specify clear what we mean by “written in a form”, we need **Regular Expression**.

It is a tool used for string matching.

For example,

[0-9]   matches any one of the character ’0’,’1’,$\cdots$,’9’

# Regular Expression



To specify clear what we mean by “written in a form”, we need **Regular Expression**.

It is a tool used for string matching.

For example,

[0-9] matches any one of the character ’0’,’1’,$\cdots$,’9’

[0-9]+ matches any string that is consists of at least one digit.

# Regular Expression

To specify clear what we mean by “written in a form”, we need **Regular Expression**.

It is a tool used for string matching.

For example,

| | |
|---|---|
| [0-9] | matches any one of the character ’0’,’1’,$\cdots$,’9’ |
| [0-9]+ | matches any string that is consists of at least one digit. |
| a*bc | matches any string that is end with “bc” and start with any number of “a’. |

## Regular Expression



To specify clear what we mean by “written in a form”, we need **Regular Expression**.

It is a tool used for string matching.

For example,

| | |
|---|---|
| [0-9] | matches any one of the character ’0’,’1’,$\cdots$,’9’ |
| [0-9]+ | matches any string that is consists of at least one digit. |
| a*bc | matches any string that is end with “bc” and start with any number of “a’. |
| [^a-z] | matches any character that is not one of ’a’,’b’,$\cdots$,’z’. |

# Regular Expression



To specify clear what we mean by “written in a form”, we need **Regular Expression**.

It is a tool used for string matching.

For example,

| | |
|---|---|
| [0-9] | matches any one of the character ’0’,’1’,$\cdots$,’9’ |
| [0-9]+ | matches any string that is consists of at least one digit. |
| a*bc | matches any string that is end with “bc” and start with any number of “a’. |
| [^a-z] | matches any character that is not one of ’a’,’b’,$\cdots$,’z’. |
| ak4\|bb10 | matches any one of the string “ak4” or “bb10”. |

## Regular Expression

To specify clear what we mean by “written in a form”, we need **Regular Expression**.

It is a tool used for string matching.

For example,

| | |
|---|---|
| [0-9] | matches any one of the character ’0’,’1’,$\cdots$,’9’ |
| [0-9]+ | matches any string that is consists of at least one digit. |
| a*bc | matches any string that is end with “bc” and start with any number of “a’. |
| [^a-z] | matches any character that is not one of ’a’,’b’,$\cdots$,’z’. |
| ak4\|bb10 | matches any one of the string “ak4” or “bb10”. |
| (ab)\|(kaab) | matches any one of the string “ab” or “kaab”[14]. |

## Regular Expression

To specify clear what we mean by “written in a form”, we need **Regular Expression**.

It is a tool used for string matching.

For example,

| | |
|---|---|
| [0-9] | matches any one of the character ’0’,’1’,· · · ,’9’ |
| [0-9]+ | matches any string that is consists of at least one digit. |
| a*bc | matches any string that is end with “bc” and start with any number of “a’. |
| [^a-z] | matches any character that is not one of ’a’,’b’,· · · ,’z’. |
| ak4\|bb10 | matches any one of the string “ak4” or “bb10”. |
| (ab)\|(kaab) | matches any one of the string “ab” or “kaab”[14]. |
| (a\|b)+ | matches any string that is consists of ’a’ and ’b’ and is non-empty. |

# Regular Expression

To specify clear what we mean by “written in a form”, we need **Regular Expression**.

It is a tool used for string matching.

For example,

| | |
|---|---|
| [0-9] | matches any one of the character ’0’,’1’,$\cdots$,’9’ |
| [0-9]+ | matches any string that is consists of at least one digit. |
| a*bc | matches any string that is end with “bc” and start with any number of “a’. |
| [^a-z] | matches any character that is not one of ’a’,’b’,$\cdots$,’z’. |
| ak4\|bb10 | matches any one of the string “ak4” or “bb10”. |
| (ab)\|(kaab) | matches any one of the string “ab” or “kaab”[14]. |
| (a\|b)+ | matches any string that is consists of ’a’ and ’b’ and is non-empty. |
| a\|b+ | matches any string “a” , “b” , “bb”, “bbb” , $\cdots$ . |

## Regular Expression

To specify clear what we mean by “written in a form”, we need **Regular Expression**.

It is a tool used for string matching.

For example,

| | |
|---|---|
| [0-9] | matches any one of the character '0','1',$\cdots$,'9' |
| [0-9]+ | matches any string that is consists of at least one digit. |
| a*bc | matches any string that is end with “bc” and start with any number of “a'. |
| [^a-z] | matches any character that is not one of 'a','b',$\cdots$,'z'. |
| ak4\|bb10 | matches any one of the string “ak4” or “bb10”. |
| (ab)\|(kaab) | matches any one of the string “ab” or “kaab”[14]. |
| (a\|b)+ | matches any string that is consists of 'a' and 'b' and is non-empty. |
| a\|b+ | matches any string “a” , “b” , “bb”, “bbb” , $\cdots$ . |
| (\+\|\-)? | matches any string “+” , “-” or $\epsilon$-string , $\cdots$ . |

## Regular Expression

To specify clear what we mean by "written in a form", we need **Regular Expression**.

It is a tool used for string matching.

For example,

| | |
|---|---|
| [0-9] | matches any one of the character '0','1',$\cdots$,'9' |
| [0-9]+ | matches any string that is consists of at least one digit. |
| a*bc | matches any string that is end with "bc" and start with any number of "a'. |
| [^a-z] | matches any character that is not one of 'a','b',$\cdots$,'z'. |
| ak4\|bb10 | matches any one of the string "ak4" or "bb10". |
| (ab)\|(kaab) | matches any one of the string "ab" or "kaab"[14]. |
| (a\|b)+ | matches any string that is consists of 'a' and 'b' and is non-empty. |
| a\|b+ | matches any string "a" , "b" , "bb", "bbb" , $\cdots$ . |
| (\+\|\-)? | matches any string "+" , "-" or $\epsilon$-string , $\cdots$ . |

[14] The parentheses here is used for grouping

# Computer Integers and Real numbers



Although we won't distinguish usually between $2.5$ and $3$, there are just number.

# Computer Integers and Real numbers



Although we won't distinguish usually between $2.5$ and $3$, there are just number. Computer would like to do so.

# Computer Integers and Real numbers

Although we won't distinguish usually between $2.5$ and $3$, there are just number.

Computer would like to do so.

It only knows whole number.

## Computer Integers and Real numbers

Although we won't distinguish usually between $2.5$ and $3$, there are just number.

Computer would like to do so.

It only knows whole number.

Computer integer is of the form (\+|\-)?[0-9]+

# Computer Integers and Real numbers

Although we won't distinguish usually between $2.5$ and $3$, there are just number.

Computer would like to do so.

It only knows whole number.

Computer integer is of the form (\+|\-)?[0-9]+

Therefore, it will treat $3/2$ to be $1$.

## Computer Integers and Real numbers

Although we won't distinguish usually between $2.5$ and $3$, there are just number.

Computer would like to do so.

It only knows whole number.

Computer integer is of the form (\+|\-)?[0-9]+

Therefore, it will treat $3/2$ to be $1$.

For simplicity, we denote [DIGIT] to be [0-9] and [INT] to be [DIGIT]+

# Computer Integers and Real numbers

Although we won't distinguish usually between $2.5$ and $3$, there are just number.

Computer would like to do so.

It only knows whole number.

Computer integer is of the form (\+|\-)?[0-9]+

Therefore, it will treat $3/2$ to be $1$.

For simplicity, we denote [DIGIT] to be [0-9] and [INT] to be [DIGIT]+

Computer treats other numbers to be computer-real number, it can be "-2.3" , "2.0", "1E+12"[15].

# Computer Integers and Real numbers

Although we won't distinguish usually between $2.5$ and $3$, there are just number.
Computer would like to do so.
It only knows whole number.

Computer integer is of the form (\+|\-)?[0-9]+
Therefore, it will treat $3/2$ to be $1$.

For simplicity, we denote [DIGIT] to be [0-9] and [INT] to be [DIGIT]+
Computer treats other numbers to be computer-real number, it can be "-2.3" , "2.0", "1E+12"[15].
Let [SIGN] $::=$ (\+|\-),

# Computer Integers and Real numbers

Although we won’t distinguish usually between $2.5$ and $3$, there are just number.

Computer would like to do so.

It only knows whole number.

Computer integer is of the form (\+|\-)?[0-9]+

Therefore, it will treat $3/2$ to be $1$.

For simplicity, we denote [DIGIT] to be [0-9] and [INT] to be [DIGIT]+

Computer treats other numbers to be computer-real number, it can be “-2.3” , “2.0”, “1E+12”[15].

Let [SIGN] $::=$ (\+|\-),

Computer real number is of the form [SIGN]?[DIGIT]*”.”[DIGIT]+((e|E)[SIGN]?[INT]) or [SIGN]?[DIGIT]+(e|E)[SIGN]?[INT].

## Computer Integers and Real numbers

Although we won't distinguish usually between $2.5$ and $3$, there are just number.

Computer would like to do so.

It only knows whole number.

Computer integer is of the form (\+|\-)?[0-9]+

Therefore, it will treat $3/2$ to be $1$.

For simplicity, we denote [DIGIT] to be [0-9] and [INT] to be [DIGIT]+

Computer treats other numbers to be computer-real number, it can be "-2.3" , "2.0", "1E+12"[15].

Let [SIGN] $::=$ (\+|\-),

Computer real number is of the form [SIGN]?[DIGIT]*"."[DIGIT]+((e|E)[SIGN]?[INT]) or [SIGN]?[DIGIT]+(e|E)[SIGN]?[INT].

Therefore, it will treat $3/2.0$ to be $1.5$ , $3.0/2$ to be $1.5$ and $3.0/2.0$ to be $1.5$.

## Computer Integers and Real numbers

Although we won't distinguish usually between $2.5$ and $3$, there are just number.

Computer would like to do so.

It only knows whole number.

Computer integer is of the form (\+|\-)?[0-9]+

Therefore, it will treat $3/2$ to be $1$.

For simplicity, we denote [DIGIT] to be [0-9] and [INT] to be [DIGIT]+

Computer treats other numbers to be computer-real number, it can be "-2.3" , "2.0", "1E+12"[15].

Let [SIGN] $::=$ (\+|\-),

Computer real number is of the form [SIGN]?[DIGIT]*"."[DIGIT]+((e|E)[SIGN]?[INT]) or [SIGN]?[DIGIT]+(e|E)[SIGN]?[INT].

Therefore, it will treat $3/2.0$ to be $1.5$ , $3.0/2$ to be $1.5$ and $3.0/2.0$ to be $1.5$.

[15] Scientific notation $1 \times 10^{12}$.

# End