1.1 Propositional Logic

New propositions can be constructed from existing propositions using *logical operators*, and are called compound propositions.

logical operators:

			other denotations:	meaning:
¬	negation	¬p	$\overline{p}$, not p	"not p"
∧	conjunction	p ∧ q	p and q	"p and q"
∨	disjunction	p ∨ q	p or q	"p or q"

example 1:
Let proposition *p* stand for "I will go to a movie theater", then ¬*p* means "I will not go to a movie theater."

Truth table for negation operation:

p	¬p
T	F
F	T

or

p	¬p
1	0
0	1

New propositions can be constructed from existing propositions using *logical operators*, and are called compound propositions.

logical operators:

			other denotations:	meaning:
¬	negation	¬p	$\overline{p}$, not p	"not p"
∧	conjunction	p ∧ q	p and q	"p and q"
∨	disjunction	p ∨ q	p or q	"p or q"

example 2:
Let proposition *p* stand for "It is raining" and *q* stand for "I want to go to a movie theater", then *p*∧*q* means "It is raining and I want to go to a movie theater."

Conjunction p ∧ q is true when both p and q are true. Truth table for p ∧ q:

p	q	p ∧ q
T	T	T
T	F	F
F	T	F
F	F	F

or

p	q	p ∧ q
1	1	1
1	0	0
0	1	0
0	0	0

1.1 Propositional Logic

New propositions can be constructed from existing propositions using *logical operators*, and are called compound propositions.

logical operators:

			other denotations:	meaning:
¬	negation	¬p	$\overline{p}$, not p	"not p"
∧	conjunction	p ∧ q	p and q	"p and q"
∨	disjunction	p ∨ q	p or q	"p or q"

example 3:
Let proposition *p* stand for "It is raining" and *q* stand for "I want to go to a movie theater", then *p*∨*q* means "It is raining or I want to go to a movie theater."

Disjunction p ∨ q is true when at least one of p and q is true. Truth table for p ∨ q:

p	q	p ∨ q
T	T	T
T	F	T
F	T	T
F	F	F

or

p	q	p ∨ q
1	1	1
1	0	1
0	1	1
0	0	0

1.1 Propositional Logic

more logical operators:

$\rightarrow$	implication	$p \rightarrow q$	" if p then q", "p implies q" *see page 6 for more*
$\oplus$	exclusive or	$p \oplus q$	"either p or q"

Implication:

$p \rightarrow q$

hypothesis, or antecedent, or premise

conclusion, or consequence

if p is true then q holds

Implication is also called conditional statement.

Truth table for the implication $p \rightarrow q$:

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

example: Let p: "the weather is good", and q: "we'll go to the beach".

Then $p \rightarrow q$ stands for "If the weather is good we'll go to the beach"

1.1 Propositional Logic

more logical operators:

$\rightarrow$	implication	$p \rightarrow q$	" if p then q", "p implies q"
$\oplus$	exclusive or	$p \oplus q$	"either p or q"

Exclusive or:

$p \oplus q$ is true when exactly one of p and q is true

example: Let p: "the weather is good", and q: "we'll go to the beach".

Then $p \oplus q$ stands for "Either the weather is good or we'll go to the beach"

Truth table for the exclusive or $p \oplus q$:

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

1.1 Propositional Logic

Precedence of Logical Operators:

$\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\oplus$

first *last*

Logic and Bit Operations

In computer all the information is represented with bits (binary digit). Bit is a symbol with two possible values: 0 (zero), and 1(one)

bit can be used to represent truth values:1 for True, 0 for False

Boolean variable is a variable with two possible values (0,1)

Note: Computer operations correspond to the logical operations.

$\wedge \equiv$ AND
$\vee \equiv$ OR
$\oplus \equiv$ XOR

x	y	$x \wedge y$	$x \vee y$	$x \oplus y$
0	0			
0	1			
1	0			
1	1			

Logic and Bit Operations

In computer all the information is represented with bits (binary digit). Bit is a symbol with two possible values: 0 (zero), and 1(one)

bit can be used to represent truth values:1 for True, 0 for False

Boolean variable is a variable with two possible values (0,1)

Note: Computer operations correspond to the logical operations.

$\wedge \equiv$ AND
$\vee \equiv$ OR
$\oplus \equiv$ XOR

x	y	$x \wedge y$	$x \vee y$	$x \oplus y$
0	0	0		
0	1	0		
1	0	0		
1	1	1		

Logic and Bit Operations

In computer all the information is represented with bits (binary digit). Bit is a symbol with two possible values: 0 (zero), and 1(one)

bit can be used to represent truth values:1 for True, 0 for False

Boolean variable is a variable with two possible values (0,1)

Note: Computer operations correspond to the logical operations.

$\wedge \equiv$ AND
$\vee \equiv$ OR
$\oplus \equiv$ XOR

x	y	$x \wedge y$	$x \vee y$	$x \oplus y$
0	0	0	0	
0	1	0	1	
1	0	0	1	
1	1	1	1	

Logic and Bit Operations

In computer all the information is represented with bits (binary digit). Bit is a symbol with two possible values: 0 (zero), and 1(one)

bit can be used to represent truth values:1 for True, 0 for False

Boolean variable is a variable with two possible values (0,1)

Note: Computer operations correspond to the logical operations.

$\wedge \equiv$ AND
$\vee \equiv$ OR
$\oplus \equiv$ XOR

x	y	$x \wedge y$	$x \vee y$	$x \oplus y$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Logic and Bit Operations

In computer all the information is represented with bits (binary digit). Bit is a symbol with two possible values: 0 (zero), and 1(one)

bit can be used to represent truth values:1 for True, 0 for False

Boolean variable is a variable with two possible values (0,1)

Note: Computer operations correspond to the logical operations.

$\wedge \equiv$ AND
$\vee \equiv$ OR
$\oplus \equiv$ XOR

x	y	$x \wedge y$	$x \vee y$	$x \oplus y$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

a bit string is a sequence of 0's and 1's.
length of a bit string is the number of bits in the string

Example: 00110101110 length=11

bit string

Exercise: find the bitwise OR, AND, and XOR of the bit strings 001110101 and 110101100 (*from now on we'll be splitting the bit strings into blocks of four to make them easier to read*).

```
0  0111  0101
1  1010  1100
-----------
```

AND

```
0  0111  0101
1  1010  1100
-----------
```

OR

```
0  0111  0101
1  1010  1100
-----------
```

XOR

x	y	x∧y	x∨ y	x⊕y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Exercise: find the bitwise OR, AND, and XOR of the bit strings 001110101 and 110101100 (*from now on we'll be splitting the bit strings into blocks of four to make them easier to read*).

```
0  0111  0101
1  1010  1100
-----------
0  0010  0100   AND
```

```
0  0111  0101
1  1010  1100
-----------
             OR
```

```
0  0111  0101
1  1010  1100
-----------
             XOR
```

x	y	x∧y	x∨ y	x⊕y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Exercise: find the bitwise OR, AND, and XOR of the bit strings 001110101 and 110101100 (*from now on we'll be splitting the bit strings into blocks of four to make them easier to read*).

```
0 0111 0101
1 1010 1100
-----------
0 0010 0100  AND
```

```
0 0111 0101
1 1010 1100
-----------
1 1111 1101  OR
```

```
0 0111 0101
1 1010 1100
-----------
           XOR
```

x	y	x∧y	x∨ y	x⊕y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Exercise: find the bitwise OR, AND, and XOR of the bit strings 001110101 and 110101100 (*from now on we'll be splitting the bit strings into blocks of four to make them easier to read*).

```
0  0111  0101
1  1010  1100
-----------
0  0010  0100   AND
```

```
0  0111  0101
1  1010  1100
-----------
1  1111  1101   OR
```

```
0  0111  0101
1  1010  1100
-----------
1  1101  1001   XOR
```

x	y	x∧y	x∨ y	x⊕y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Exercise: Evaluate expression (0 1101 $\vee$ 1001) $\oplus$ (01101 $\wedge$ 00111)

```
0 1101          0 1101
  1001          0 0111
------          ------          ------
   OR              AND              ⊕
```

x	y	x∧y	x∨ y	x⊕y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Exercise: Evaluate expression (0 1101 $\vee$ 1001) $\oplus$ (01101 $\wedge$ 00111)

```
0 1101          0 1101
  1001          0 0111
------          ------          ------
0 1101 OR       0 0101 AND             ⊕
```

x	y	x∧y	x∨ y	x⊕y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Exercise: Evaluate expression (0 1101 $\vee$ 1001) $\oplus$ (01101 $\wedge$ 00111)=0 100

```
0 1101          0 1101          0 1101
  1001          0 0111          0 0101
------          ------          ------
0 1101 OR       0 0101 AND      0 1000 ⊕
```

x	y	x∧y	x∨ y	x⊕y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0