Logic

X. Zhang,
Fordham Univ.

Motivating example

- Four machines A, B, C, D are connected on a network. It is feared that a computer virus may have infected the network. Your security team makes the following statements:
  - If D is infected, then so is C.
  - If C is infected, then so is A.
  - If D is clean, then B is clean but C is infected.
  - If A is infected, then either B is infected or C is clean.
- Based on these statements, what can you conclude about status of the four machines?

Smullyan's Island Puzzle

- You meet two inhabitants of Smullyan's Island (where each one is either a liar or a truth-teller).
  - A says, "Either B is lying or I am"
  - B says, "A is lying"
- Who is telling the truth ?

Symbolic logic

- Subjects: statements that is either true or false, i.e., propositions
- Understand relations between statements
  - Equivalent statement: can we simplify and therefore understand a statement better ?
  - Contradictory statements: can both statements be true ?
  - Reasoning: does a statement follow from a set of hypothesis ?
- Application: solve logic puzzle, decide validity of reasoning/proof …

Roadmap

- Simple Proposition
- Logic operations & compound proposition
  - Unary operation: negation
  - Binary operation: conjuction (AND) , disjuction (OR), conditional ($\Rightarrow$) , biconditional ($\Leftrightarrow$)
  - Evaluating order & truth table
- Tautology, contradiction, equivalence
- Logic operation laws
- Applications: solving logic puzzles

Proposition

- Proposition: a statement which is either true or false
  - For example:
    - Ten is less than seven.
    - There are life forms on other planets in the universe.
    - A set of cardinality n has 2^n subsets.
  - The followings are not propositions:
    $$x^2 = 16$$
    - How are you ?
    - x+y<10

Proposition

- If a proposition is true, we say it has a truth value of true; otherwise, we say it has a truth value of false.
- a lower case letter is used to represent a proposition
  - Let p stands for “Ten is smaller than seven”
  - p has truth value of false, i.e., F.
- Analogy to numerical algebra
  - Variables represented by letters
  - Possible values for variables are {T, F}

Compound Proposition

- One can connect propositions using “and”, “or”, “not”, “only if” …to form compound proposition:
  - It will not rain tomorrow.
  - Fishes are jumping and the cotton is high.
  - If the ground is wet then it rains last night.
- Truth value of compound proposition depends on truth value of the simple propositions
  - We will formalize above connectives as operations on propositions

Negation

- It will not rain tomorrow. $\neg p$
  - It's not the true that it will rain tomorrow.
  - It's false that it will rain tomorrow. p
- Negation ($\neg$) applies to a single proposition
  - If p is true, then $\neg p$ is false
  - If p is false, then $\neg p$ is true
- We can use a table to summarize :

p	$\neg p$
T	F
F	T

All possible values of the input

$\neg p$, output/function values

Truth table

- Truth table: a table that defines a logic operation or function, i.e., allow one to look up the function's value under given input values

p	$\neg p$
T	F
F	T

All possible values of the input

$\neg p$, output/function values

Logical Conjunction (AND, $\wedge$)

- To say two propositions are both true:
  - Peter is tall and thin.
  - The hero is American and the movie is good.
- The whole statement is true if both simple propositions are true; otherwise it's false.
  - We use $\wedge$ (read as "and") to denote logic conjuction:

t	h	$t \wedge h$
T	T	T
T	F	F
F	T	F
F	F	F

Recognizing conjunction connectives

- English words connecting the propositions might be "but", "nevertheless", "unfortunately", …. For example:
  - Although the villain is French, the movie is good. $v \wedge g$
  - The hero is not American, but the villain is French. $(\neg h) \wedge v$
- As long as it means that both simple propositions are true, it's an AND operation.

Practice

- Introduce letters to stand for simple propositions, and write the following statements as compound propositions:
  - It's sunny and cold.
  - The movie is not long, but it's very interesting.

Different meaning of "OR"

- "… or …, but not both".
  - You may have coffee or you may have tea.
  - Mike is at the tennis court or at the swimming pool.
- "… or …, or both".
  - The processor is fast or the printer is slow.
- To avoid confusion:
  - By default we assume the second meaning, unless it explicitly states "not both".

Exclusive Or

- Exclusive or ($\oplus$) : exactly one of the two statements is true, cannot both be true
  - I will watch movies or read a book tonight, but not both.
  - You may have coffee or you may have tea, but not both.
  - Mike is at the tennis court or at the swimming pool.

c	**d**	$c \oplus d$
T	T	F
T	F	T
F	T	T
F	F	F

Logical Disjunction (Inclusive Or)

- Inclusive or ($\vee$) : at least one of the two statements is true (can be both true)
- The processor is small or the memory is small.
  - "The process is small" (p) or "The memory is small" (m), denoted as
- Truth table for inclusive or: $p \vee m$

p	m	$p \vee m$
T	T	T
T	F	T
F	T	T
F	F	F

Outline

- Simple Proposition
- Logic operations & compound proposition
  - Unary operation: negation
  - Binary operation: conjuction (AND) , disjuction (OR), conditional ($\Rightarrow$) , biconditional ($\Leftrightarrow$)
  - Evaluating order & truth table
- Logic equivalence
  - Logic operation laws
- Applications: solving logic puzzles

Logic Connection: implication/ *conditional*

- Some compound propositions states logical connection between two simple propositions (rather than their actual truthfulness)
  - If it rains, then the ground is wet.
- Logic implication statement has two parts:
  - If part: hypothesis
  - Then part: conclusion
  - If the hypothesis is true, then the conclusion is true.
  - use $\Rightarrow$ to connect hypothesis and conclusion
- logic implication is called *conditional* in textbook

Truth table for logic implication

- "If I am elected, then I will lower the taxes next year".
  - e: I am elected.
  - l: I lower the taxes next year.
  - i.e., if e is true, then l must be true.
  - We use $e \Rightarrow l$ to denote this compound statement.

e	l	$e \Rightarrow l$
T	T	T
T	**F**	**F**
F	T	T
F	F	T

Understand logic implication

e	l	$e \Rightarrow l$
T	T	T
T	F	F
F	T	T
F	F	T

- Under what conditions, the promise is broken, i.e., the statement is false ?
  - When I am elected, but I do not lower the taxes next year.
- For all other scenarios, I keep my promise, i.e., the statement is true.
  - I am elected, and lower the taxes next year
  - I am not elected, I lower the taxes next year.
  - I am not elected, I do not lower the taxes next year.

Many different English Expressions

- In spoken language, there are many ways to express implication (if … then…)
  - It rains, therefore the ground is wet.
- Wet ground follows rain.
- As long as it rains, the ground is wet.
- Rain is a sufficient condition for the ground to be wet.
- When translating English to proposition forms
- Rephrase sentence to “if …. Then…” without change its meaning

Example: from English to Proposition form

- Write the following in proposition form:
  - A British heroine is a necessary condition for the movie to be good.
  - b: "The heroine is a British".
  - m: "The movie is good"
  - The heroine needs/has to be a British for the movie to be true.
  - If the movie is good, then the heroine is a British.
  - So the propositional form is

$$m \Rightarrow b$$

Write following in propositional forms:

- If the movie is good, only if the hero is American.
- A good diet is a necessary condition for a healthy cat.
- A failed central switch is a sufficient condition for a total network failure.

Some exercises

- Purchasing a lottery ticket is a _______ condition for winning the lottery.
  - Winning the lottery is a _______ condition for purchasing a lottery ticket.
- You have to take the final exam in order to pass the CISC1100 course.
  - Taking the final exam is a _______ condition of passing CISC1100.
  - Passing CISC1100 is a ________ condition of taking the final exam.

Outline

- Simple Proposition
- Logic operations & compound proposition
  - Unary operation: negation
  - Binary operation: conjuction (AND) , disjuction (OR), conditional ($\Rightarrow$) , biconditional ($\Leftrightarrow$)
  - Evaluating order & truth table
- Tautology, contradiction, equivalent
- Logic operation laws
- Applications: solving logic puzzles

Complicated propositions

- Connectives can be applied to compound propositions, e.g.:

$$\neg(p \wedge q) \qquad (\neg p) \vee (p \wedge q)$$

p	q	$p \wedge q$	$\neg(p \wedge q)$
T	T	T	F
T	F	F	T
F	T	F	T
F	F	F	T

- The order of evaluation (book P. 43)

Writing truth table : $(\neg p) \vee (p \wedge q)$

- First fill in all possible input values
  - For 2 variables, p, q, there are 4 possible input values:

p	q	$\neg p$	$p \wedge q$	$(\neg p) \vee (p \wedge q)$
T	T			
T	F			
F	T			
F	F			

- Next, create a column for each compound propositions, $\neg p$ $p \wedge q$ $(\neg p) \vee (p \wedge q)$
- Third, fill in the columns one by one, starting from simple ones

Input values

- For a propositions with n variables
  - There are 2^n possible input value combinations, i.e., 2^n rows for the truth table
- Use the following pattern to enumerate all input value combinations
  - The last variable follows TFTF… pattern (1)
  - The second last variable: TTFFTTFF… pattern (2)
  - The third last: TTTTFFFFTTTTFFFF... (4)
  - The fourth last: TTTTTTTTFFFFFFFF … (8)
  - …

An example

- For a form with 3 simple propositions

$$(\neg p) \wedge (q \wedge r)$$

p	q	r	$\neg p$	$q \wedge r$	$(\neg p) \wedge (q \wedge r)$
T	T	T	F	T	F
T	T	F	F	F	F
T	F	T	F	F	F
T	F	F	F	F	F
F	T	T	T	T	T
F	T	F	T	F	F
F	F	T	T	F	F
F	F	F	T	F	F

Practice

- Introduce letters to stand for simple propositions, and write the following statements as compound propositions:
  - The food is good or the service is excellent.

    $$g \vee s$$

  - Neither the food is good nor the service is excellent.

    $$\neg g \wedge \neg s$$

  - He is good at math, or his girl friend is and helps him.

    $$g \vee (f \wedge h)$$

Sufficient and necessary condition

- Examples:
  - Lighting is sufficient and necessary condition for thunder.

$$(l \Rightarrow t) \wedge (t \Rightarrow l)$$

  - The knight will win if and only if the armor is strong.
    - The knight will win if the armor is strong. $s \Rightarrow w$
    - The knight will win only if the armor is strong. $w \Rightarrow s$

$$(s \Rightarrow w) \wedge (w \Rightarrow s)$$

Biconditional connective

$$p \Leftrightarrow q := (p \Rightarrow q) \wedge (q \Rightarrow p)$$

- p if and only if q,
- p is sufficient and necessary condition of q

p	q	$p \Rightarrow q$	$q \Rightarrow p$	$p \Leftrightarrow q$
T	T	T	T	**T**
T	F	F	T	**F**
F	T	T	F	**F**
F	F	T	T	**T**

Precedence Rules

- Parenthesized subexpressions come first
- Precedence hierarchy
  - Negation ($\neg$) comes next
  - *Multiplicative* operations ($\wedge$) is done before *additive* operations ($\vee, \oplus$)
  - Conditional-type operations ($\Rightarrow, \Leftrightarrow$) are done last
- In case of a tie, operations are evaluated in left-to-right order, except for conditional operator ($\Rightarrow, \Leftrightarrow$) which is evaluated in right-to-left order
  - $p \vee q \oplus r$ is evaluated as $(p \vee q) \oplus r$
  - $p \Rightarrow q \Rightarrow r$ is evaluated as $p \Rightarrow (q \Rightarrow r)$

Outline

- Simple Proposition
- Logic operations & compound proposition
  - Unary operation: negation
  - Binary operation: conjuction (AND) , disjuction (OR), conditional ($\Rightarrow$) , biconditional ($\Leftrightarrow$)
  - Evaluating order & truth table
- Propositional equivalence
  - Propositional identities
- Applications: solving logic puzzles

Logical equivalence

- Two propositional forms **are logically equivalent**, if they have same truth value under all conditions
  - Analogous to algebra rules

$$p \vee q \text{ and } q \vee p$$

$$p \Rightarrow q \text{ and } \neg p \vee q$$

- We represent logical equivalence using $\equiv$

$$p \Rightarrow q \equiv \neg p \vee q$$

- To prove or disprove logical equivalency
  - Draw and Compare true tables of the two forms

Logic Identities (1)

- Commutative
  - 1. $p \wedge q \equiv q \wedge p$
  - 2. $p \vee q \equiv q \vee p$
- Associative
  - 1. $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
  - 2. $(p \vee q) \vee r \equiv p \vee (q \vee r)$

Logic Identities (2)

- **DeMorgan's laws**
  - 1. $\neg(p \wedge q) \equiv \neg p \vee \neg q$
  - 2. $\neg(p \vee q) \equiv \neg p \wedge \neg q$

Logic Identities (3)

- **Distributive**
  - 1. $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
  - 2. $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$

Logic Identities (4)

- **Double negative**

$$\neg(\neg p) \equiv p$$

- **Contrapositive**

$$(p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$$

Simplify propositional forms

- Human beings understand the simplified form much better…
  - Put negation closer to simple proposition
  - Get rid of double negation
- Simplify following propositional forms, i.e., find a simpler equivalent form

$$p \vee (\neg p \wedge q)$$

$$\equiv (p \vee \neg p) \wedge (p \vee q) \text{ using distributive law}$$

$$\equiv T \wedge (p \vee q) \text{ p is either true or false, so } p \vee \neg p \text{ is True}$$

$$\equiv p \vee q$$

- Key: apply logical equivalence rules such as DeMorgan Law, implication law, double negation …

Simplify propositional forms (2)

- Key: apply logical equivalence rules such as DeMorgan Law, implication law, double negation …
  - We don't know how to directly negate a "if … then" form
  - First apply implication law, then use DeMorgan law:

$$\neg(p \Rightarrow \neg q)$$

$$\equiv \neg(\neg p \vee \neg q) \text{ implication law}$$

$$\equiv \neg\neg p \wedge \neg\neg q$$

$$\equiv p \wedge q$$

Outline

- Simple Proposition
- Logic operations & compound proposition
  - Unary operation: negation
  - Binary operation: conjuction (AND) , disjuction (OR), conditional ($\Rightarrow$) , biconditional ($\Leftrightarrow$)
  - Evaluating order & truth table
- Propositional equivalence
  - Propositional identities
- Applications: solving logic puzzles

Solving problem using logic

- Four machines A, B, C, D are connected on a network. It is feared that a computer virus may have infected the network. Your security team makes the following statements:
  - If D is infected, then so is C.
  - If C is infected, then so is A.
  - If D is clean, then B is clean but C is infected.
  - If A is infected, then either B is infected or C is clean.

Solving problem using logic

- Four machines A, B, C, D are connected on a network. It is feared that a computer virus may have infected the network. Your security team makes the following statements:
  - If D is infected, then so is C.
  - If C is infected, then so is A.
  - If D is clean, then B is clean but C is infected.
  - If A is infected, then either B is infected or C is clean.
- How many possibilities are there ?
  1. A, B,C, D are all be clean
  2. A, B,C are clean, D is infected,
  3. A,B,D are clean, C is infected,
  4. ….

  …
- Is the first case possible ? The second ? …

Application of Logic

A case study

An example

- Your friend's comment:
  - If the birds are flying south and the leaves are turning, then it must be fall. Falls brings cold weather. The leaves are turning but the weather is not cold. **Therefore** the birds are not flying south.
- Is her argument sound/valid?

An example

- Is her argument sound/valid?
  - Suppose the followings are true:
    - If the birds are flying south and the leaves are turning, the it must be fall.
    - Falls brings cold weather.
    - The leaves are turning but the weather is not cold.
  - Can one conclude “the birds are not flying south” ?

Reasoning & Proving

- Prove by contradiction
  - Assume the birds are flying south,
  - then since leaves are turning too, then it must be fall.
  - Falls bring cold weather, so it must be cold.
  - But it’s actually not cold.
  - We have a contradiction, therefore our assumption that the birds are flying south is wrong.

Indirect Proofs (Section 3.1.8)

- Direct Proof vs Indirect Proof
- Show that the square of an odd number is also an odd number
  - Suppose that m is an odd number,
  - We will show that m^2 is also an odd number.
  - Direct Proof: (from givens to the conclusion)
    - Let $m=2n+1$ (since m is an odd number)
    - Then $m^2=(2n+1)^2=4n^2+4n+1=2(2n^2+2n)+1$
    - Therefore m^2 is an odd number

Indirect Proof technique

- Proof by contradiction:
  - Rather than proving the conclusion is true, we assume that it is false, and see whether this assumption leads to some kind of contradiction.
  - A contradiction would mean the assumption is wrong, i.e., the conclusion is true

Indirect Proof technique

- Show that square of an odd number is also an odd number
  - Suppose that m is an odd number,
  - We will show that m^2 is also an odd number.
  - Indirect Proof:
    - Assume m^2 is an even number
    - As $m^2=m*m$, m must be an even number. This contradicts with the fact that m is an odd number.
    - Therefore the assuming cannot be true, and therefore m^2 is odd

Underlying logic laws

- We can prove the following logic equivalence

$$p \Rightarrow q \equiv q' \Rightarrow p'$$

- Therefore, in order to prove $p \Rightarrow q$, we prove $q' \Rightarrow p'$

Next: application to computer

Other Positional Numeral System

- Base: number of digits (symbols) used in the system.
  - Base 2 (i.e., binary): only use 0 and 1
  - Base 8 (octal): only use 0,1,…7
  - Base 16 (hexadecimal): use 0,1,…9, A,B,C,D,E,F
- Like in decimal system,
  - Rightmost digit: represents its value times the base to the zeroth power
  - The next digit to the left: times the base to the first power
  - The next digit to the left: times the base to the second power
  - …
  - For example: binary number 10101

$= 1*2^4+0*2^3+1*2^2+0*2^1+1*2^0=16+4+1=21$

Why binary number?

- Computer uses binary numeral system, i.e., base 2 positional number system
  - Each unit of memory media (hard disk, tape, CD …) has two states to represent 0 and 1
  - Such physical (electronic) device is easier to make, less prone to error
    - E.g., a voltage value between 0-3mv is 0, a value between 3-6 is 1 …

Binary => Decimal

- Interpret binary numbers (transform to base 10)
  - 1101

$= 1*2^3+1*2^2+0*2^1+1*2^0=8+4+0+1=13$

- Translate the following binary number to decimal number
  - 101011

Generally you can consider other bases

- Base 8 (Octal number)
  - Use symbols: 0, 1, 2, …7
  - Convert octal number 725 to base 10:

    $=7*8^2+2*8^1+5=\ldots$
  - Now you try:

  $(1752)_8 =$
- Base 16 (Hexadecimal)
  - Use symbols: 0, 1, 2, …9, A, B, C,D,E, F
  - $(10A)_{16} = 1*16^2+10*16^0=..$

Binary number arithmetic

- Analogous to decimal number arithmetics
- How would you perform addition?
  - 0+0=0
  - 0+1=1
  - 1+1=10 (a carry-over)
  - Multiple digit addition: 11001+101=
- Subtraction:
  - Basic rule:
  - Borrow one from next left digit

From Base 10 to Base 2: using table

- Input : a decimal number
- Output: the equivalent number in base 2
- Procedure:
  - Write a table as follows
  1. Find the largest two's power that is smaller than the number
     1. Decimal number 234 => largest two's power is 128
  2. Fill in 1 in corresponding digit, subtract 128 from the number => 106
  3. Repeat 1-2, until the number is 0
  4. Fill in empty digits with 0

…	512	256	128	64	32	16	8	4	2	1
			1	1	1	0	1	0	1	0

  - Result is 11101010

From Base 10 to Base 2: the recipe

- Input : a decimal number
- Output: the equivalent number in base 2
- Procedure:
  1. Divide the decimal number by 2
  2. Make the remainder the next digit to the left of the answer
  3. Replace the decimal number with the quotient
  4. If quotient is not zero, Repeat 1-4; otherwise, done

Convert 100 to binary number

100 % 2 = 0

=> last digit

100 / 2 = 50

50 % 2 = 0

=> second last digit

50/2 = 25

25 % 2 = 1

=> 3rd last digit

25 / 2 = 12

The result is 1100100

12 % 2 = 0 => 4th last digit

12 / 2 = 6

6 % 2 = 0 => 5th last digit

6 / 2 = 3

3 % 2 = 1

=> 6th last digit

3 / 2 =1

1 % 2 = 1

=> 7th last digit

1 / 2 = 0

Stop as the decimal # becomes 0

Data Representation in Computer

- In modern computers, all information is represented using binary values.
- Each storage location (cell): has two states
  - low-voltage signal => 0
  - High-voltage signal => 1
  - i.e., it can store a binary digit, i.e., bit
- Eight bits grouped together to form a byte
- Several bytes grouped together to form a word
  - Word length of a computer, e.g., 32 bits computer, 64 bits computer

Different types of data

- Numbers
  - Whole number, fractional number, …
- Text
  - ASCII code, unicode
- Audio
- Image and graphics
- video

How can they all be represented as binary strings?

Representing Numbers

- Positive whole numbers
  - We already know one way to represent them: i.e., just use base 2 number system
- All integers, i.e., including negative integers
  - Set aside a bit for storing the sign
    - 1 for +, 0 for –
- Decimal numbers, e.g., 3.1415936, 100.34
  - Floating point representation:
    - sign * mantissa * 2 ^{exp}
  - 64 bits: one for sign, some for mantissa, some for exp.

Representing Text

- Take English text for example
- Text is a series of characters
  - letters, punctuation marks, digits 0, 1, …9, spaces, return (change a line), space, tab, …
- How many bits do we need to represent a character?
  - 1 bit can be used to represent 2 different things
  - 2 bit … $2*2 = 2^2$ different things
  - n bit 2^n different things
- In order to represent 100 diff. character
  - Solve $2^n = 100$ for n
  - n = $\lceil \log_2 100 \rceil$, here the $\lceil x \rceil$ refers to the ceiling of x, i.e., the smallest integer that is larger than x:

$$\lceil \log_2 100 \rceil = \lceil 6.6438 \rceil = 7$$

There needs a standard way

- ASCII code: **American Standard Code for Information Interchange**
  - ASCII codes represent text in computers, communications equipment, and other devices that use text.
  - 128 characters:
    - 33 are non-printing control characters (now mostly obsolete) [7] that affect how text and space is processed
    - 94 are printable characters
    - space is considered an invisible graphic

ASCII code

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

There needs a standard way

- Unicode
  - international/multilingual text character encoding system, tentatively called Unicode
  - Currently: 21 bits code space
  - How many diff. characters?
- Encoding forms:
  - UTF-8: each Unicode character represented as one to four 8-but bytes
  - UTF-16: one or two 16-bit code units
  - UTF-32: a single 32-but code unit

How computer processing data?

- Through manipulate digital signals (high/low)
- Using addition as example

  10000111

  + 0001110

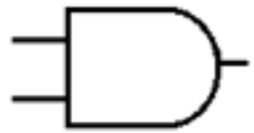
- Input: the two operands, each consisting of 32 bits (i.e., 32 electronic signals)
- Output: the sum
- How ?

Digital Logic

- Performs operation on one or more logic inputs and produces a single logic output.
- Can be implemented
  - electronically using diodes or transistors
  - Using electromagnetic relays
  - Or other: fluidics, optics, molecules, or even mechanical elements
- We won't go into the physics of how is it done, instead we focus on the input/output, and logic

Basic building block

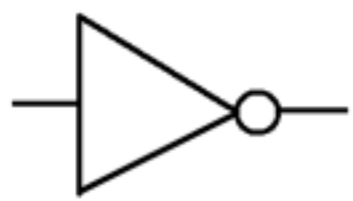
- Basic Digital logic is based on primary functions (the basic gates):
  - AND



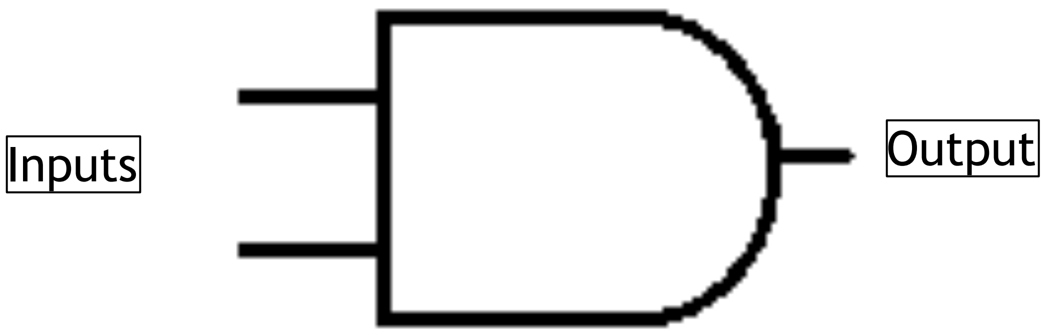
  - OR

  - XOR

  - NOT



AND Logic Symbol

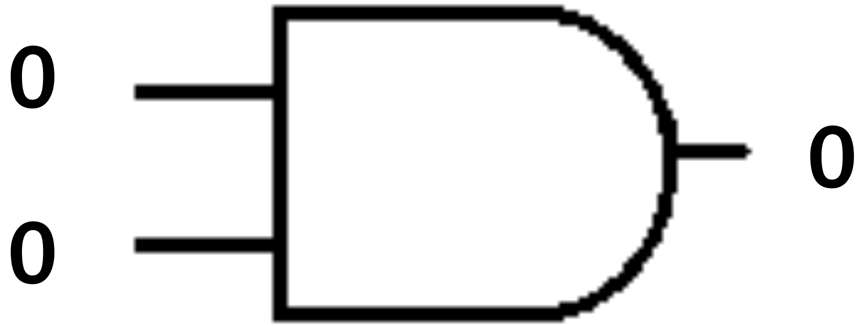


If both inputs are 1, the output is 1

If any input is 0, the output is 0

AND Logic Symbol

Inputs



Output

Determine the output

Animated Slide

AND Logic Symbol

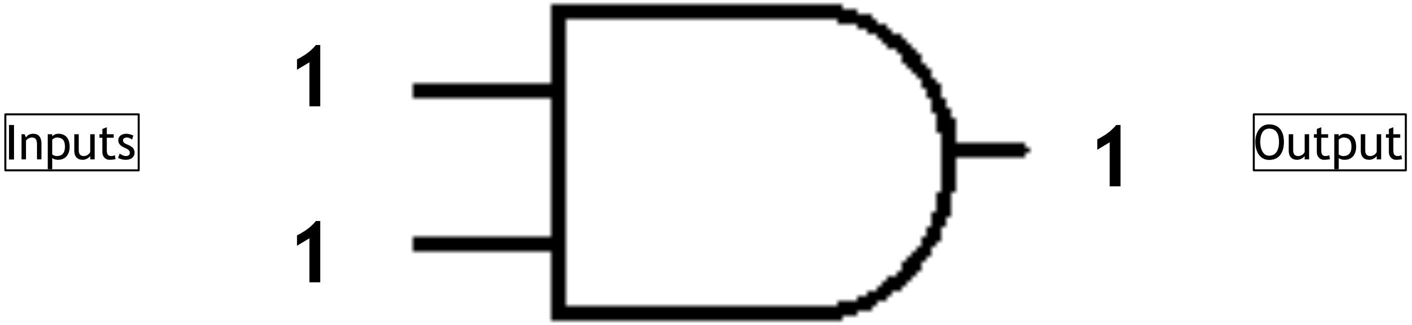
Inputs

Output

Determine the output

Animated Slide

AND Logic Symbol



Determine the output

Animated Slide

AND Truth Table

- To help understand the function of a digital device, a Truth Table is used:

Every possible input combination

Input		Output
0	0	0
0	1	0
1	0	0
1	1	1

AND Function

OR Logic Symbol

Inputs

Output

If any input is 1, the output is 1

If all inputs are 0, the output is 0

OR Logic Symbol

Inputs

0

0

0

Output

Determine the output

Animated Slide

OR Logic Symbol

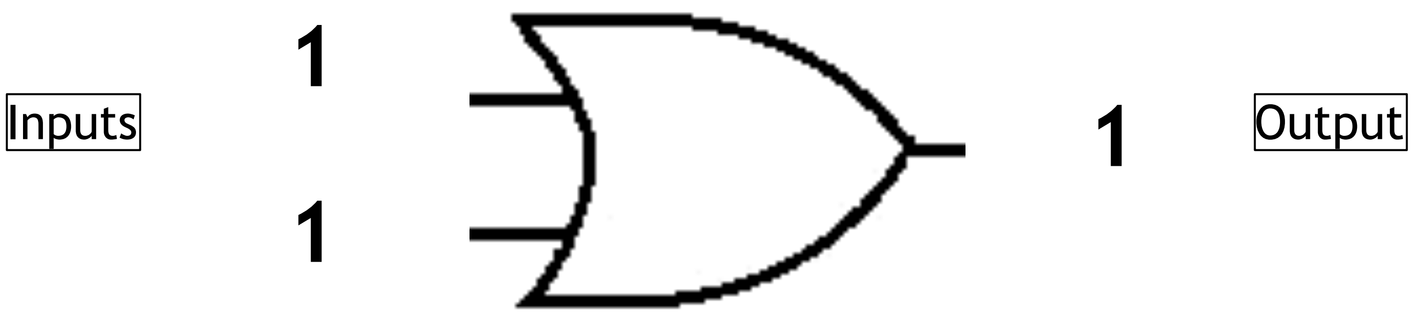
Inputs

Output

Determine the output

Animated Slide

OR Logic Symbol



Determine the output

Animated Slide

OR Truth Table

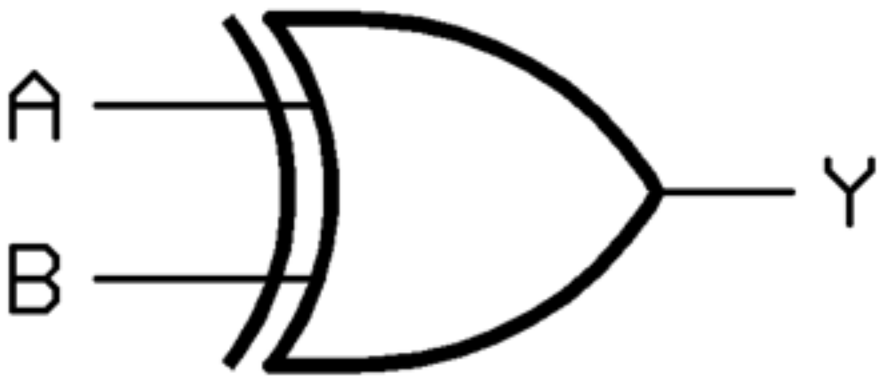
- Truth Table

Input		Output
0	0	0
0	1	1
1	0	1
1	1	1

OR Function

XOR Gate

- The XOR function:
  - if exactly one input is high, the output is high
  - If both inputs are high, or both are low, the output is low



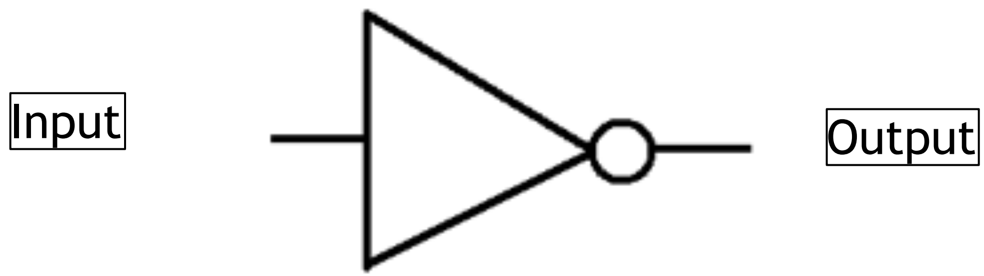
XOR Truth Table

- Truth Table

Input		Output
0	0	0
0	1	1
1	0	1
1	1	0

XOR Function

NOT Logic Symbol

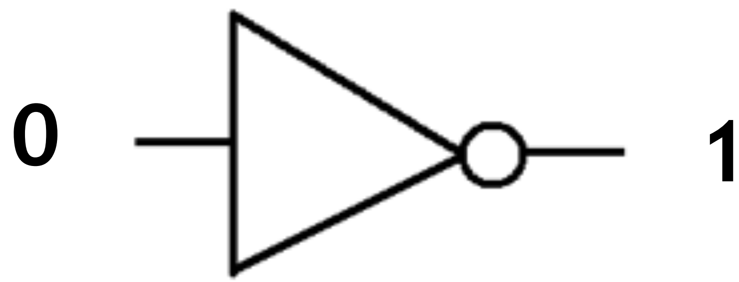


If the input is 1, the output is 0

If the input is 0, the output is 1

NOT Logic Symbol

Input



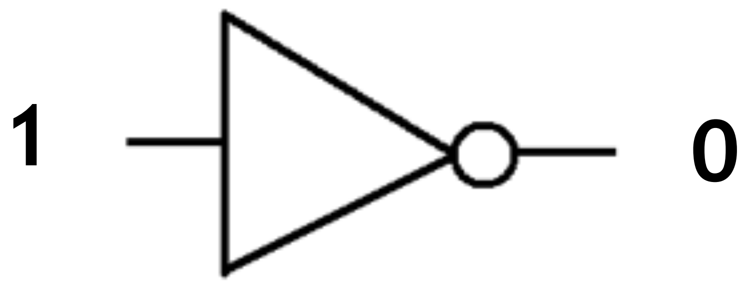
Output

Determine the output

Animated Slide

NOT Logic Symbol

Input



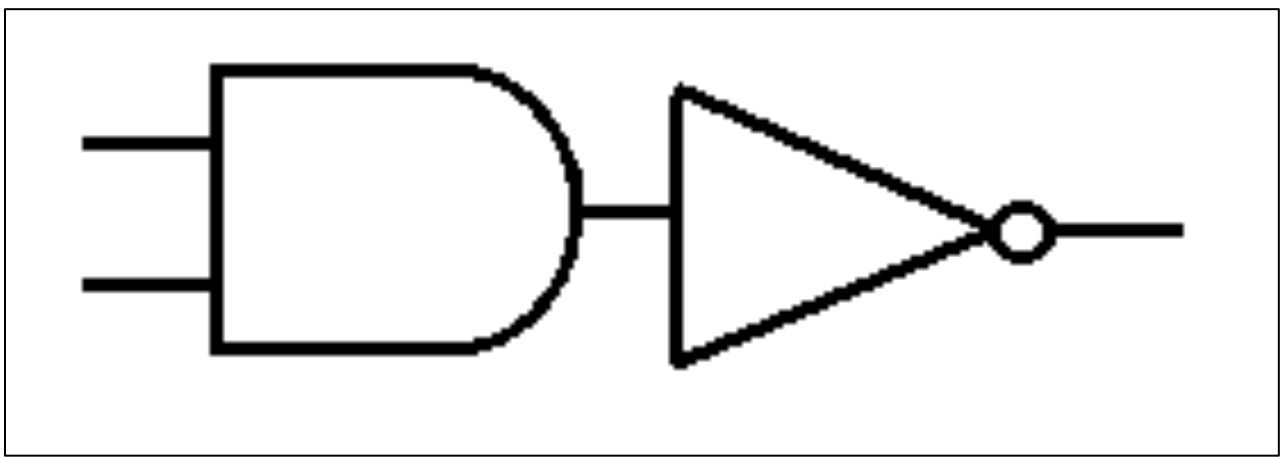
Output

Determine the output

Animated Slide

Combinational logic

- A circuit that utilizes more that one logic function has Combinational Logic.
  - How would your describe the output of this combinational logic circuit?

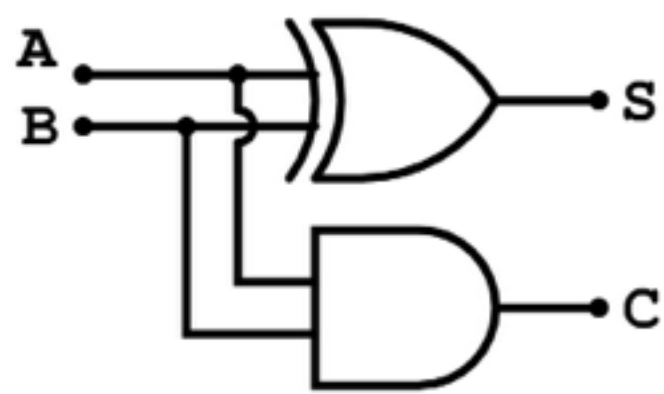


Combinational Logic: Half Adder

- Electronic circuit for performing single digit binary addition

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Sum
- Carry = A AND B



Full Adder

- One bit full adder with carry-in and carry-out

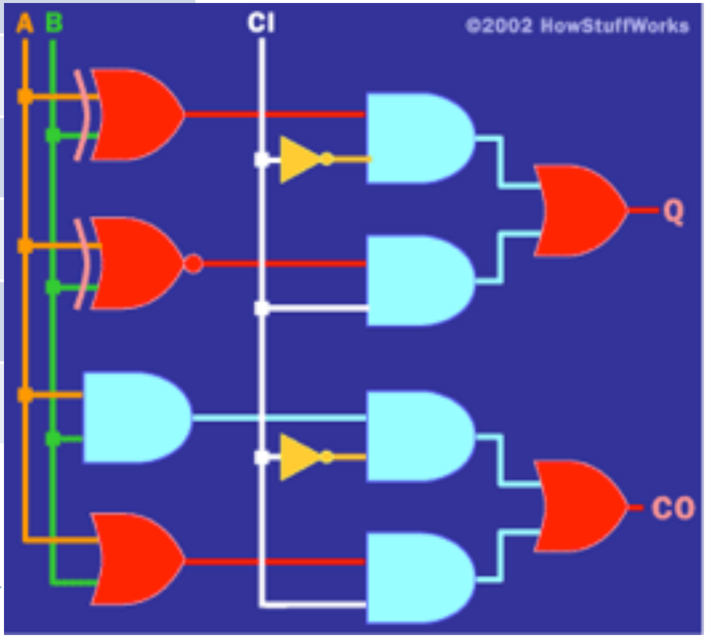
A	B	CI	Q	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

$p \vee (\neg p \wedge q)$

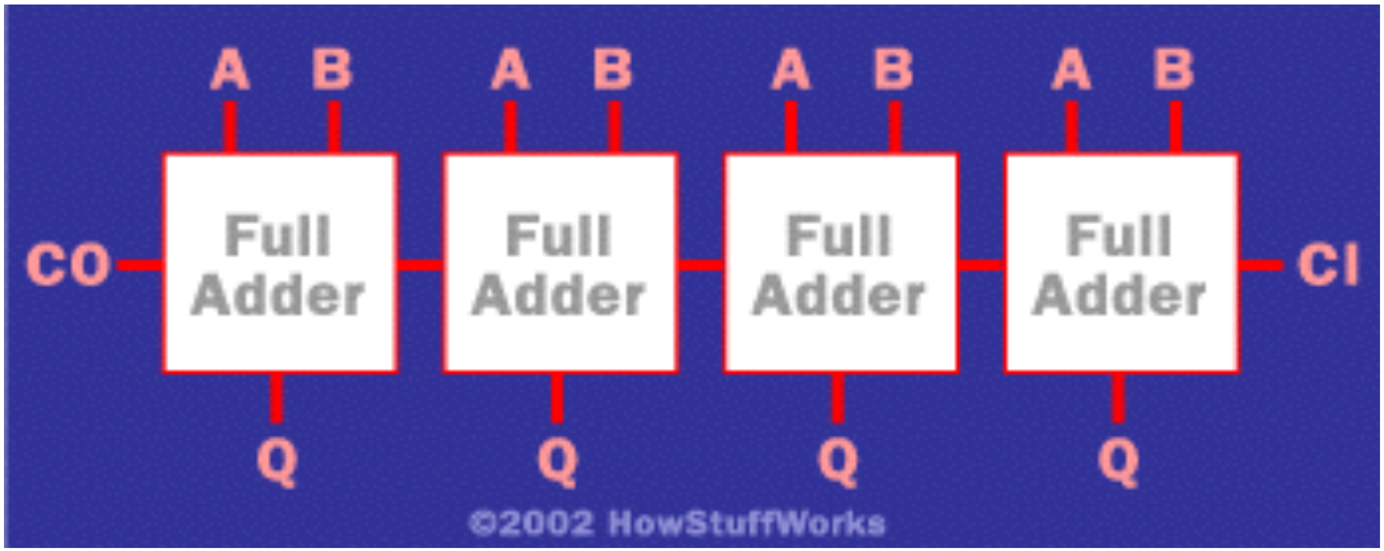
$\equiv (p \vee \neg p) \wedge (p \vee q)$ using distributive law

$\equiv T \wedge (p \vee q)$ p is either true or false, so $p \vee \neg p$ is True

$\equiv p \vee q$



Full adder: 4-digit binary addition



Chain 4 full-adders together, lower digit's carry-out is fed into the higher digit as carry-in

Integrated Circuit

- Also called chip
  - A piece of silicon on which

multiple gates are embedded

  - mounted on a package with pins, each pin is either an input, input, power or ground
- Classification based on # of gates
  - VLSI (Very Large-Scale Integration) > 100,000 gates

CPU (Central Processing Unit)

- Many pins to connect to memory, I/O
- Instruction set: the set of machine instructi

supported by an architecture (such as Pentiu

  - Move data (between register and memory)
  - Arithmetic Operations
  - Logic Operations:
  - Floating point arithmetic
  - Input/Output