COMP 102: Computers and Computing
Lecture 3: Truth Tables and Logic Gates

Instructor: Kaleem Siddiqi (siddiqi@cim.mcgill.ca)

Class web page: www.cim.mcgill.ca/~siddiqi/102.html

Practice example

- Three friends are trying to decide what to do Saturday night (see a movie or go out clubbing). They settle the issue by a vote (everyone gets a single vote, the activity with the most votes wins.)

- Assume you want a computer to automatically compile the votes and declare the winning activity.

- What <u>logical variables</u> would you use?

- Can you write a <u>logical expression</u>, which evaluates whether or not you will go Clubbing (True = Clubbing, False = Movie)?

Practice example

- Input logical variables:
  - V1 = Vote of person 1 (True=Clubbing, False=Movie)
  - V2 = Vote of person 2 (True=Clubbing, False=Movie)
  - V3 = Vote of person 3 (True=Clubbing, False=Movie)

- Output logical variables
  - ACTIVITY = Choice of activity (True=Clubbing, False=Movie)

- Logical expression:

  ACTIVITY = (V1 AND V2) OR (V1 AND V3) OR (V2 AND V3)

How would you check if the logical expression is correct?

Checking logical expressions

- Computer must be ready for <u>any</u> input, and must compute <u>correct</u> results in all cases.

- Must go through all possible input combinations:
  - V1=True, V2=True, V3=True ACTIVITY = ?
  - V1=True, V2=True, V3=False ACTIVITY = ?
  - V1=True, V2=False, V3=True ACTIVITY = ?
  - V1=True, V2=False, V3=False ACTIVITY = ?
  - V1=False, V2=True, V3=True ACTIVITY = ?
  - V1=False, V2=True, V3=False ACTIVITY = ?
  - V1=False, V2=False, V3=True ACTIVITY = ?
  - V1=False, V2=False, V3=False ACTIVITY = ?

Truth table

- Write-up a table with all possible input combinations, and check the output the output for each row.

Inputs:			Outputs:
V1	V2	V3	ACTIVITY
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

- This is called a **Truth Table**.

Comparing logical expressions

- Recall our previous expression:

  ACTIVITY = (V1 AND V2) OR (V1 AND V3) OR (V2 AND V3)

- You can also extract the logical expression directly from the Truth Table:

  ACTIVITY = ((NOT V1) AND V2 AND V3) OR
  (V1 AND (NOT V2) AND V3) OR
  (V1 AND V2 AND (NOT V3)) OR
  (V1 AND V2 AND V3)

Extracting logical expression from the truth table

- Recall:

  ACTIVITY = ((NOT V1) AND V2 AND V3) OR
  (V1 AND (NOT V2) AND V3) OR
  (V1 AND V2 AND (NOT V3)) OR
  (V1 AND V2 AND V3)

- How do we get this logical expression:
  - Consider each line in the table.
    - If the line has OUTPUT=1, this line must be included in the logical expression as a sub-expression.
    - The sub-expression includes all variables, where true variables are included without modification and negative variables are preceded by NOT operator.
    - The variables in a sub-expression are separated by “AND” logical operators.
  - Sub-expressions are separated by “OR” logical operators.
  - This is a “disjunction” of “conjunctions”.

Pros / cons of the two logical expressions

- Compare:

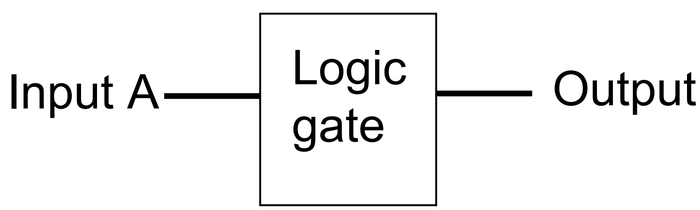
E1: ACTIVITY = (V1 AND V2) OR (V1 AND V3) OR (V2 AND V3)

E2: ACTIVITY = ((NOT V1) AND V2 AND V3) OR
(V1 AND (NOT V2) AND V3) OR
(V1 AND V2 AND (NOT V3)) OR
(V1 AND V2 AND V3)

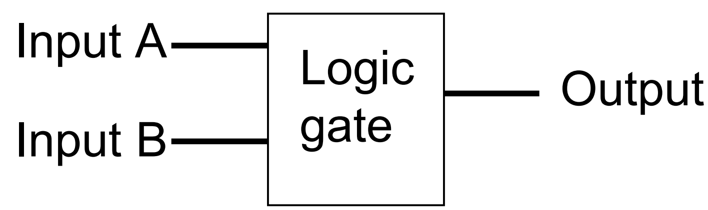
- E1 is more compact.
- E2 we can get directly from the truth table.

How do we implement a logical expression?

- Assume we have logic gates (or blocks) that implement each logical operator.



Single input



Double input

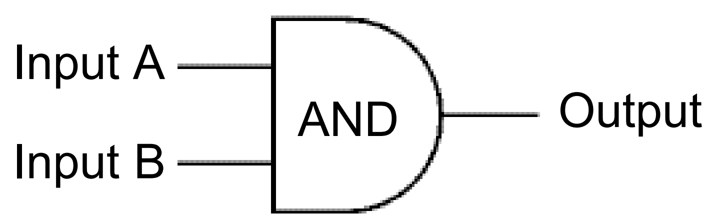
- Logic gates are the building blocks of digital electronics and are used to build telecommunication devices, computers, etc.

Logic gates and their truth table: AND

- Truth table for the AND operator:

Input A:	Input B:	Output:
0	0	0
0	1	0
1	0	0
1	1	1

- The AND gate is usually drawn as a half-moon.
- Below is a <u>two input</u> version.

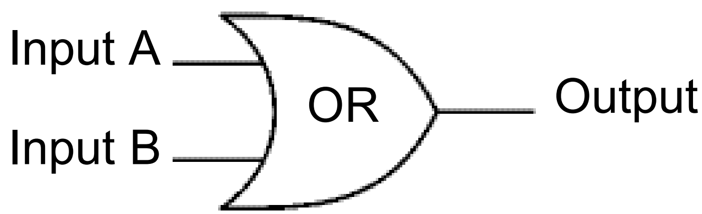


Logic gates and their truth table: OR

- Truth table for the OR operator:

Input A:	Input B:	Output:
0	0	0
0	1	1
1	0	1
1	1	1

- The OR gate is usually drawn as a crescent-shape.
- Below is a two input version.

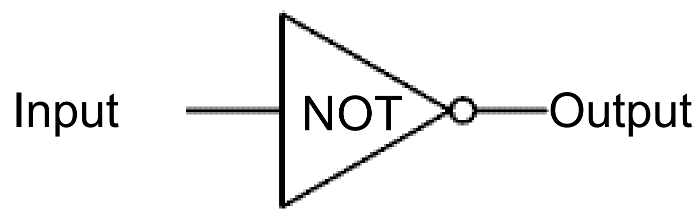


Logic gates and their truth table: NOT

- Truth table for the NOT operator:

Input:	Output:
0	1
1	0

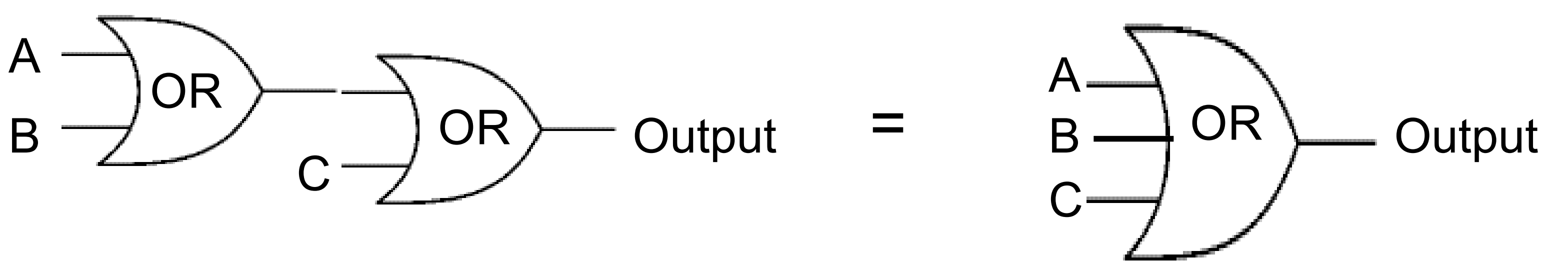
- The NOT gate is usually drawn as a triangle, with a small circle.
- The circle is there to indicate the output inverts the input.
- It is a <u>single input</u> gate.



Can we implement E1 and E2?

- Problem:
  - E1 needs a 3-input OR gate.
  - E2 needs a 3-input AND gate.

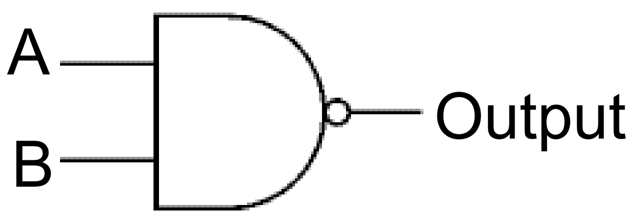
- Solution: Make it by grouping gates



- And similarly for AND gates.

More complicated gates: NAND

- NAND = Not AND
- This corresponds to an AND gate followed by a NOT gate.
  - Output = NOT (A AND B)

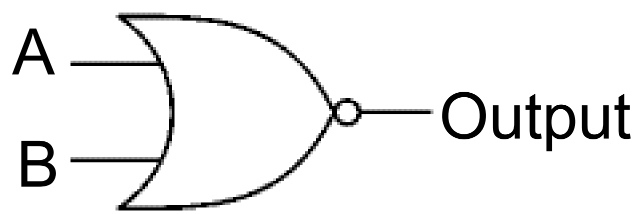


Input A	Input B	Output Q
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table

More complicated gates: NOR

- NOR = Not OR
- This corresponds to an OR gate followed by a NOT gate.
  - Output = NOT (A OR B)

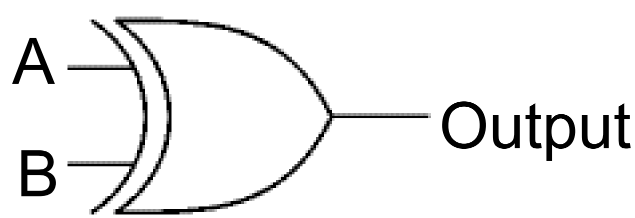


Input A	Input B	Output Q
0	0	1
0	1	0
1	0	0
1	1	0

Truth Table

More complicated gates: EX-OR

- EX-OR = EXclusive OR
- This corresponds to an OR gate, except that the output is FALSE when both inputs are TRUE.



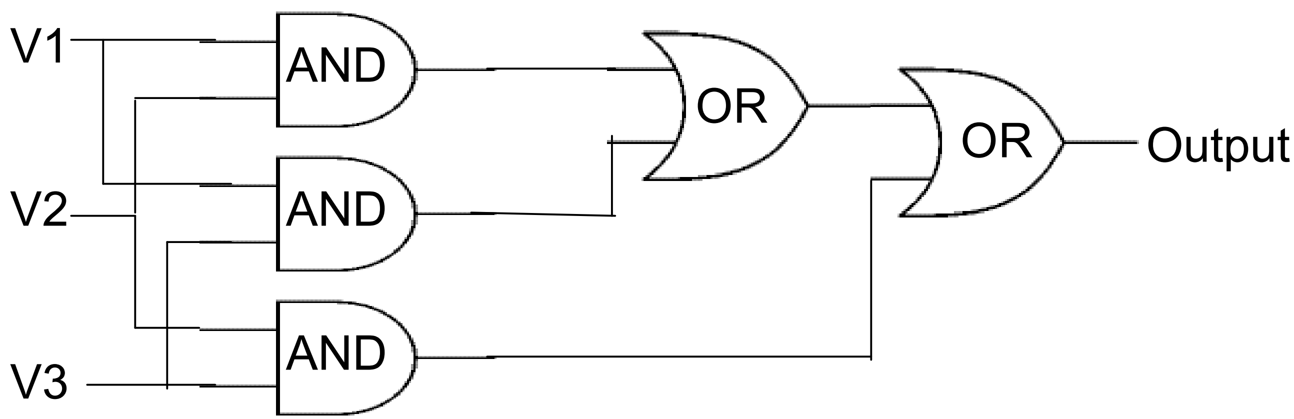
Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table

Combining logic gates

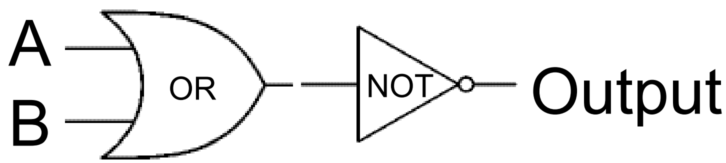
- Logic gates can be combined to produce complex logical expressions.

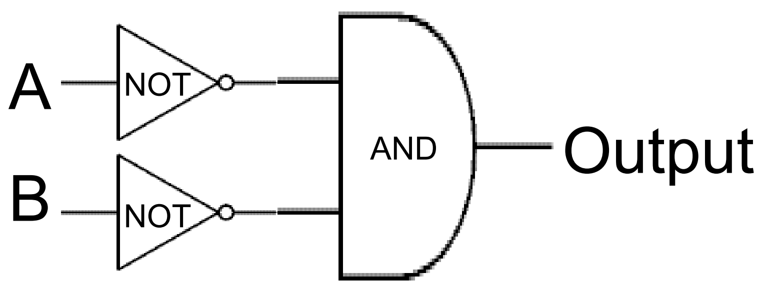
  E.g.: ACTIVITY = (V1 AND V2) OR (V1 AND V3) OR (V2 AND V3)



- Logic gates can also be combined to substitute for another type of gate.

Example



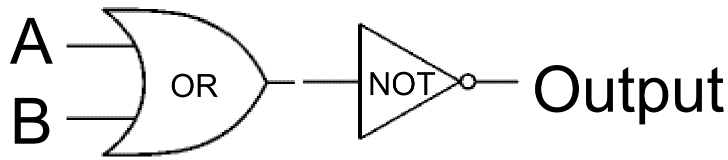


Is there a unique set of blocks to represent a given expression? No!

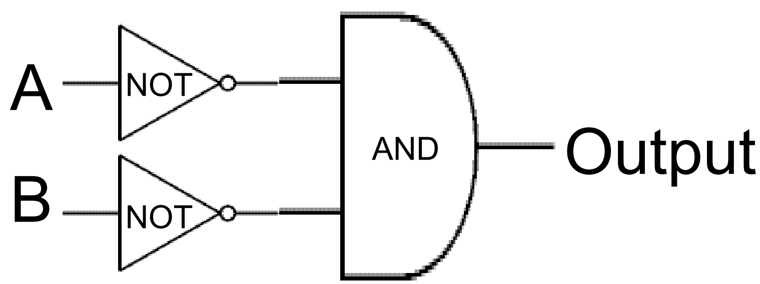
(Hint: Just write out the truth table for each set of gates, and see whether they are the same.)

De Morgan's Theorem

<u>Logical gates</u> | <u>Logical expression</u>



NOT (A OR B)

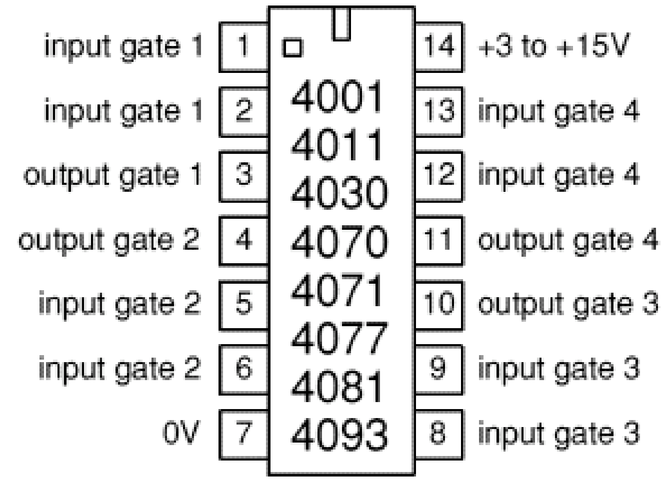


(NOT A) AND (NOT B)

Theorem: If neither A nor B is true, then both A and B must be false.

How do we choose which expression to implement?

- Sometimes function can be more compact (recall E1 vs E2).
- Multiple logic gates (of one type) are placed on a single chip; it may be more efficient to use all of them, rather than require another chip (of a different type).



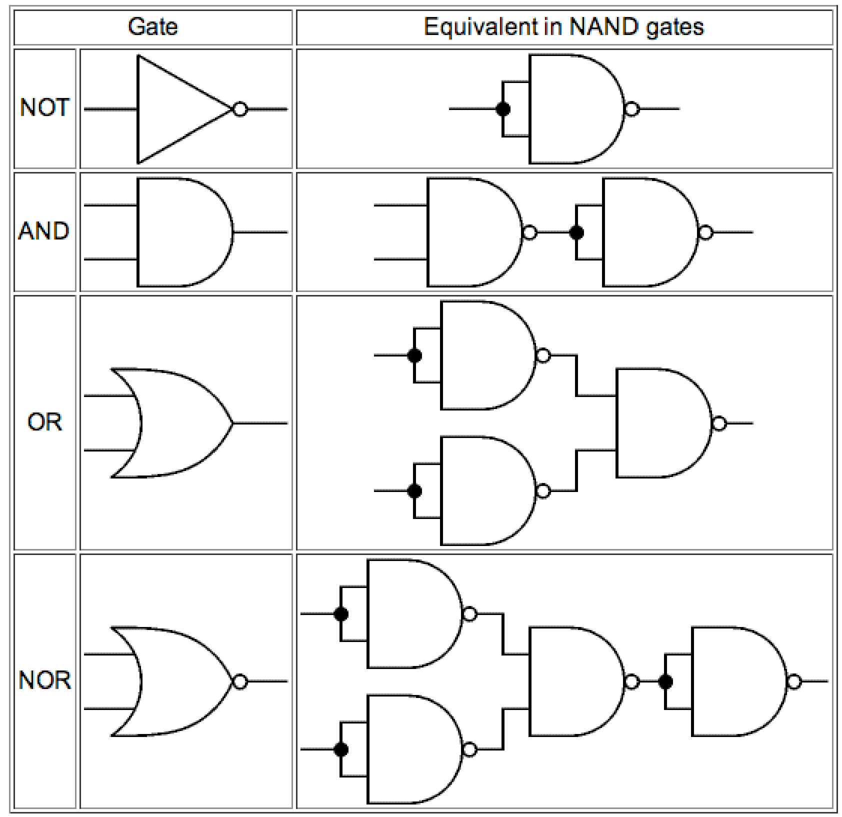
4001 Chip: four 2-input NOR gates

Leveraging this insight

- Any logical gate can be replaced by a <u>set of NAND gates</u>.

- This means all we ever need, to implement any logical expression, is a lot of NAND gates!

- Total number of gates may be larger, but total number of chips is usually smaller.



(Note: Same thing can be done with NOR gates.)

A slightly harder problem

- Imagine you play a game of **Rock-Paper-Scissors** against your friend.
- Assume you want a computer to automatically decide if you win or not.
- What logical variables would you use?
- Can you write a logical expression, which evaluates whether or not you win (True = win, False = loose)?
  - E.g. If you play Rock and your friend players Scissor, it returns True, and similarly for other possible plays.

Rock-Paper-Scissors: Logical variables

- Input: choice of player 1, choice of player 2
- Output: outcome of the game (according to the rules)
- Need to convert input and output to <u>binary</u> representation.
- Need 2 variables to represent the possible choice of each player

  01 = Scissors   10 = Paper   11 = Rock

  So we need 4 variables to represent the choice of both players.
- Need 2 variables to represent the possible outcomes.

  10 = Player 1 wins   01 = Player 2 wins = 10   00 = Tie

Rock-Paper-Scissors: Other representations

- There are other possible binary representations.

- Some are equivalent:
  - same expressive power, same number of bits
  - E.g. Scissors = 00, Paper = 01, Rock = 11

- Some are not equivalent:
  - E.g. Scissors = 0, Paper = 1, Rock = 1 (Fewer bits, less expressive power)
  - E.g. Scissors = 000, Paper = 001, Rock = 011 (Same power, but more bits)

Rock-Paper-Scissors: Truth Table

Input logical variables: Player1:	A	B	Player2:	C	D	**Output variables:**	E	F
Scissors	0	1	Scissors	0	1	Tie	0	0
Scissors	0	1	Paper	1	0	Player 1 wins	1	0
Scissors	0	1	Rock	1	1	Player 2 wins	0	1
Paper	1	0	Scissors	0	1	Player 2 wins	0	1
Paper	1	0	Paper	1	0	Tie	0	0
Paper	1	0	Rock	1	1	Player 1 wins	1	0
Rock	1	1	Scissors	0	1	Player 1 wins	1	0
Rock	1	1	Paper	1	0	Player 2 wins	0	1
Rock	1	1	Rock	1	1	Tie	0	0

What happens to the unspecified input (e.g. 0000)? Doesn’t matter what the output is!

Rock-Paper-Scissors: Logical expressions

- Need two expressions, one for each of the output bits.

E = ((NOT A) AND B AND C AND (NOT D)) OR

((A AND (NOT B) AND C AND D)) OR

((A AND B AND (NOT C) AND D))

F = ((NOT A) AND B AND C AND D) OR

(A AND (NOT B) AND (NOT C) AND D) OR

(A AND B AND C AND (NOT D))

Rock-Paper-Scissors: Logical gates

- Final step! Try this at home.

Take-home message

- Know how to build a truth table from a logical problem description.
- Know how to extract the logical expressions from the truth table.
- Learn to identify and use the basic gates: AND, OR, NOT.
- Understand the link between truth tables and logic gates.
- Know how to use combinations of gates to implement logical expressions.
- Understand that many different sets of gates can represent a given logical expression.
- Be able to state and understand De Morgan's theorem.

Final comments

- Some material from these slides was taken from:
  - *http://www.cs.rutgers.edu/~mlittman/courses/cs442-06/*
  - *http://www.kpsec.freeuk.com/gates.htm*