31268 Web Systems

UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Information Technology

Week 08: Computer Science 2
Part 2: Advanced Logic

Mathematics and Computer Science

UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Information Technology

- **Under the bonnet**
  - Representation of information
  - Number systems
- **Logic and Mathematics**
  - Boolean Algebra
  - Binary Arithmetic
- **Storage and processing of information**
  - Computation
  - Memory
  - Coding

Objectives and Motivation

You must learn to:

- Write logical expressions
- Understand when two logical expressions mean the same thing

In order to:

- Translate between your reasoning and the computer's reasoning.
- Find the best way to write code and make chips!

Laws of Boolean logic

- http://en.wikipedia.org/wiki/Laws_of_classical_logic
- Associativity
- Distributivity
- Complement/Identity
- Commutativity
- De Morgans

Boolean Associativity

- AND and OR are both associative operations
  i.e. order is not important for same operations

(x AND y) AND z = x AND (y AND z)
(x OR y) OR z = x OR (y OR z)

- Therefore, one may write
  (x AND y) AND z

as

  x AND y AND z

- But DO NOT mix the ANDs and ORs!

Boolean Associativity

- AND and OR are both associative operations
  i.e. order is not important for same operations

(x AND y) AND z = x AND (y AND z)
(x OR y) OR z = x OR (y OR z)

- Therefore, one may write
  (x AND y) AND z

as

  x AND y AND z

- But DO NOT mix the ANDs and ORs!

looks like arithmetic rules?
eg: (a * b) * c = a * (b * c)

Boolean Distributivity

It can be proven that:

- AND distributes over OR:

(x OR y) AND z = (x AND z) OR (y AND z)

- OR distributes over AND:

(x AND y) OR z = (x OR z) AND (y OR z)

Boolean Distributivity

It can be proven that:

Looks like arithmetic?
eg: (a + b) * c
= (a * c) + (b * c)

- AND distributes over OR:

(x OR y) AND z = (x AND z) OR (y AND z)

- OR distributes over AND:

(x AND y) OR z = (x OR z) AND (y OR z)

Boolean Distributivity

Observation 1

x AND y OR z

is not valid notation, since

(x AND y) OR z

is not equal to

x AND (y OR z)

e.g. Let x=FALSE, y=FALSE, z=TRUE
(FALSE and FALSE) or TRUE ➔ TRUE
FALSE and (FALSE or TRUE) ➔ FALSE

Boolean Distributivity

UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Information Technology

Observation 2

Natural numbers are **less** distributive:

- ✓ Logic: (x OR y) AND z = (x AND z) OR (y AND z)
- ✓ Math: e.g. (1 + 1) * 2 **=** (1 * 2) + (1 * 2)

but

- ✓ Logic: (x AND y) OR z = (x OR z) AND (y OR z)
- × Math: (1*1) + 2 **!=** (1 + 2)*(1 + 2)

Boolean Distributivity

UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Information Technology

Observation 2

Natural numbers are **less** distributive:

- ✓ Logic: (x OR y) AND z = (x AND z) OR (y AND z)
- ✓ Math: e.g. (1 + 1) * 2 **=** (1 * 2) + (1 * 2)

but

we can almost treat AND as *
Also OR as +

- ✓ Logic: (x AND y) OR z = (x OR z) AND (y OR z)
- × Math: (1*1) + 2 **!=** (1 + 2)*(1 + 2)

Complement/Identity Law

- X *OR TRUE* = TRUE
- X *OR FALSE* = X

- X AND TRUE = X
- X AND FALSE = FALSE
  → this implies X AND not(X) = FALSE
  - (also called non-Contradiction law!)

de Morgan's Laws

- not (P) OR not (Q) = not (P AND Q)
- not (P) AND not (Q) = not (P OR Q)

Verify

- By experimenting with examples from daily life
- By constructing a table for each composite operation.

de Morgan's Laws

- not (P) OR not (Q) = not (P AND Q)
- not (P) AND not (Q) = not (P OR Q)

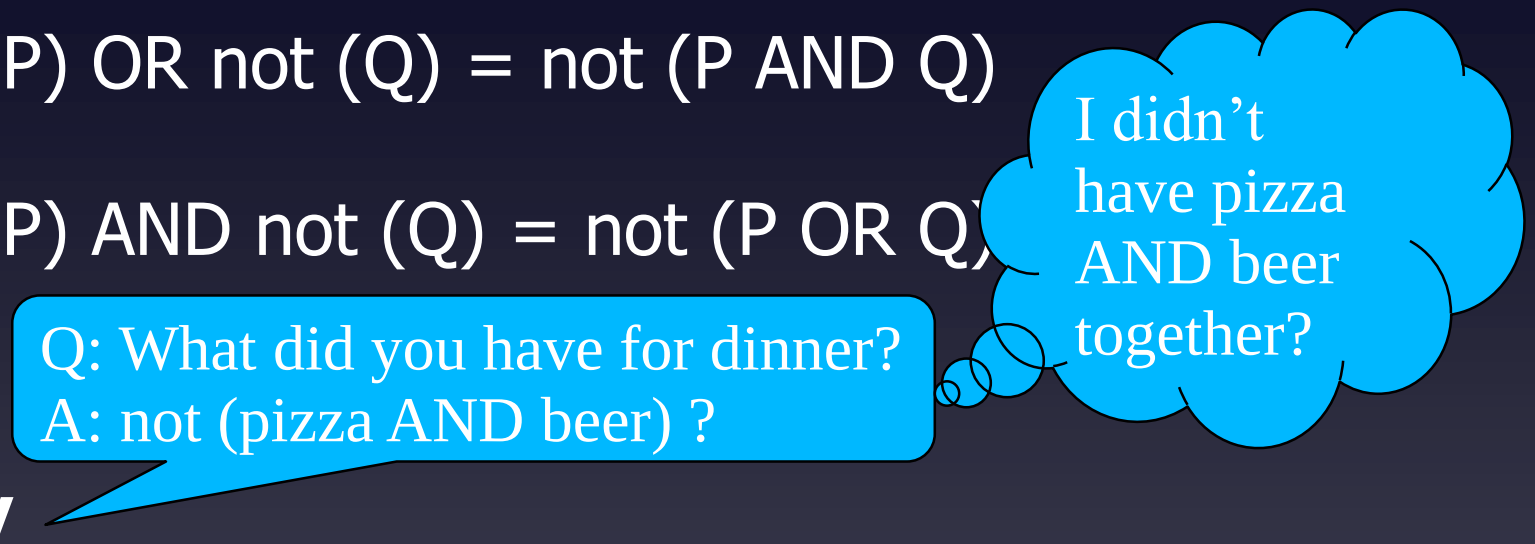
Q: What did you have for dinner?
A: not (pizza AND beer) ?

Verify

- By experimenting with examples from daily life
- By constructing a table for each composite operation.

de Morgan's Laws

- not (P) OR not (Q) = not (P AND Q)
- not (P) AND not (Q) = not (P OR Q)



Verify

- By experimenting with examples from daily life
- By constructing a table for each composite operation.

de Morgan's Laws

- not (P) OR not (Q) = not (P AND Q)
- not (P) AND not (Q) = not (P OR Q)

Q: What did you have for dinner?
A: not (pizza AND beer) ?

I didn't have pizza AND beer together?

Verify

- By experimenting with examples from daily life
- By constructing a table for each composite operation.

ie not(pizza) or not(beer)
I MIGHT have had pizza, or MIGHT have had beer **BUT NOT BOTH TOGETHER!**

Reasons for de Morgan's Laws

- Allow **simplification** of complex Boolean expressions
- Can be used to prove that a NAND gate with negated Inputs is the equivalent of an OR gate.

Reasons for de Morgan's Laws

- Allow **simplification** of complex Boolean expressions
- Can be used to prove that a NAND gate with negated Inputs is the equivalent of an OR gate.
- Allows Integrated Circuits (IC) to be produced with fewer gates to perform the same functions.
  - We only need NOT and NAND gates to do all functions!
  - CHEAPER to build one set of gates
  - Many modern IC now just consist of an array of NAND gates. Millions (if not Billions) of these gates. Easier to manufacture!
- Designing IC becomes mainly an exercise in making the right connections between NAND gates to produce logic functions.

Put into practice

- What does A or Not(B and A) = ?

Put into practice

- What does A or Not(B and A) = ?

- A or (Not(B) or Not(A))
  using DeMorgan's Law's

Put into practice

- What does A or Not(B and A) = ?
- A or (Not(B) or Not(A))
- (A or Not(A)) or Not(B)
  using Associativity

Put into practice

- What does A or Not(B and A) = ?

- A or (Not(B) or Not(A))

- (A or Not(A)) or Not(B)

- True or Not(B)
  using Complement Law

Put into practice

- What does A or Not(B and A) = ?

- A or (Not(B) or Not(A))

using DeMorgan's Law's

- (A or Not(A)) or Not(B)

using Associativity

- True or Not(B)

- **True**

using Complement law

Questions?

- You will be required to know these laws to simplify logic in the exam