

# Introduction to Logic

**Sections 1.1, 1.2, 1.3 of Rosen**

Spring 2013

CSCS 235 Introduction to Discrete Structures

URL: [cse.unl.edu/~cse235](http://cse.unl.edu/~cse235)

All questions: [Piazza](#)

# Introduction: Logic?

---

- We will study
  - Propositional Logic (PL)
  - First-Order Logic (FOL)
- Logic
  - is the study of the logic relationships between objects and
  - forms the basis of all mathematical reasoning and all automated reasoning

# Introduction: PL?

---

- Propositional Logic (PL) = Propositional Calculus = Sentential Logic
- In Propositional Logic, the objects are called propositions
- **Definition:** A proposition is a statement that is either true or false, but not both
- We usually denote a proposition by a letter:  $p$ ,  $q$ ,  $r$ ,  $s$ , ...

# Outline

---

- Defining Propositional Logic
  - Propositions
  - Connectives
  - Precedence of Logical Operators
  - Truth tables
- Usefulness of Logic
  - Bitwise operations
  - Logic in Theoretical Computer Science (SAT)
  - Logic in Programming
- Logical Equivalences
  - Terminology
  - Truth tables
  - Equivalence rules

# Introduction: Proposition

---

- **Definition:** The value of a proposition is called its truth value; denoted by
  - $T$  or 1 if it is true or
  - $F$  or 0 if it is false
- Opinions, interrogative, and imperative are not propositions
- **Truth table**

$p$
0
1

# Propositions: Examples

---

- The following are propositions
  - Today is Monday *M*
  - The grass is wet *W*
  - It is raining *R*
- The following are not propositions
  - C++ is the best language *Opinion*
  - When is the pretest? *Interrogative*
  - Do your homework *Imperative*

# Are these propositions?

---

- $2+2=5$
- Every integer is divisible by 12
- Microsoft is an excellent company

# Logical connectives

---

- Connectives are used to create a compound proposition from two or more propositions
  - Negation (e.g.,  $\neg a$  or  $!a$  or  $\bar{a}$ )  $\backslashneg$, \bar$$
  - And or logical conjunction (denoted  $\wedge$ )  $\wedge$$
  - OR or logical disjunction (denoted  $\vee$ )  $\vee$$
  - XOR or exclusive or (denoted  $\oplus$ )  $\oplus$$
  - Implication (denoted  $\Rightarrow$  or  $\rightarrow$ )  $\Rightarrow$, \rightarrow$$
  - Biconditional (denoted  $\Leftrightarrow$  or  $\leftrightarrow$ )  $\Leftrightarrow$, \leftrightarrow$$
- We define the meaning (semantics) of the logical connectives using truth tables



# Precedence of Logical Operators

---

- As in arithmetic, an ordering is imposed on the use of logical operators in compound propositions
- However, it is preferable to use parentheses to disambiguate operators and facilitate readability

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

- To avoid unnecessary parenthesis, the following precedences hold:
  1. Negation ( $\neg$ )
  2. Conjunction ( $\wedge$ )
  3. Disjunction ( $\vee$ )
  4. Implication ( $\rightarrow$ )
  5. Biconditional ( $\leftrightarrow$ )

# Logical Connective: Negation

---

- $\neg p$ , the negation of a proposition  $p$ , is also a proposition
- Examples:
  - Today is not Monday
  - It is not the case that today is Monday, etc.
- **Truth table**

$p$	$\neg p$
0	1
1	0

# Logical Connective: Logical And

---

- The logical connective And is true only when both of the propositions are true. It is also called a conjunction
- Examples
  - It is raining and it is warm
  - $(2+3=5)$  and  $(1<2)$
  - Schroedinger's cat is dead and Schroedinger's cat is not dead.
- Truth table

$p$	$q$	$p \wedge q$
0	0	
0	1	
1	0	
1	1	

# Logical Connective: Logical OR

---

- The logical disjunction, or logical OR, is true if one or both of the propositions are true.
- Examples
  - It is raining or it is the second lecture
  - $(2+2=5) \vee (1<2)$
  - You may have cake or ice cream

- **Truth table**

$p$	$q$	$p \wedge q$	$p \vee q$
0	0	0	
0	1	0	
1	0	0	
1	1	1	

# Logical Connective: Exclusive Or

---

- The exclusive OR, or XOR, of two propositions is true when exactly one of the propositions is true and the other one is false
- Example
  - The circuit is either ON or OFF but not both
  - Let  $ab < 0$ , then either  $a < 0$  or  $b < 0$  but not both
  - You may have cake or ice cream, but not both

- Truth table

$p$	$q$	$p \wedge q$	$p \vee q$	$p \oplus q$
0	0	0	0	
0	1	0	1	
1	0	0	1	
1	1	1	1	

# Logical Connective: Implication (1)

---

- **Definition:** Let  $p$  and  $q$  be two propositions. The implication  $p \rightarrow q$  is the proposition that is false when  $p$  is true and  $q$  is false and true otherwise
  - $p$  is called the hypothesis, antecedent, premise
  - $q$  is called the conclusion, consequence

- **Truth table**

$p$	$q$	$p \wedge q$	$p \vee q$	$p \oplus q$	$p \Rightarrow q$
0	0	0	0	0	
0	1	0	1	1	
1	0	0	1	1	
1	1	1	1	0	

# Logical Connective: Implication (2)

---

- The implication of  $p \rightarrow q$  can be also read as
  - If  $p$  then  $q$
  - $p$  implies  $q$
  - If  $p, q$
  - $p$  **only** if  $q$
  - $q$  if  $p$
  - $q$  when  $p$
  - $q$  whenever  $p$
  - $q$  follows from  $p$
  - $p$  is a **sufficient** condition for  $q$  ( $p$  is sufficient for  $q$ )
  - $q$  is a **necessary** condition for  $p$  ( $q$  is necessary for  $p$ )

# Logical Connective: Implication (3)

---

- Examples
  - If you buy you air ticket in advance, it is cheaper.
  - If  $x$  is an integer, then  $x^2 \geq 0$ .
  - If it rains, the grass gets wet.
  - If the sprinklers operate, the grass gets wet.
  - If  $2+2=5$ , then all unicorns are pink.



# Exercise: Which of the following implications is true?

---

- If  $-1$  is a positive number, then  $2+2=5$

True. The premise is obviously false, thus no matter what the conclusion is, the implication holds.

- If  $-1$  is a positive number, then  $2+2=4$

True. Same as above.

- If you get an 100% on your Midterm 1, then you will have an  $A^+$  on CSCE235

False. Your grades homework, quizzes, Midterm 2, and Final, if they are bad, would prevent you from having an  $A^+$ .

# Logical Connective: Biconditional (1)

---

- **Definition:** The biconditional  $p \leftrightarrow q$  is the proposition that is true when  $p$  and  $q$  have the same truth values. It is false otherwise.
- Note that it is equivalent to  $(p \rightarrow q) \wedge (q \rightarrow p)$

- **Truth table**

$p$	$q$	$p \wedge q$	$p \vee q$	$p \oplus q$	$p \Rightarrow q$	$p \leftrightarrow q$
0	0	0	0	0	1	
0	1	0	1	1	1	
1	0	0	1	1	0	
1	1	1	1	0	1	

# Logical Connective: Biconditional (2)

---

- The biconditional  $p \leftrightarrow q$  can be equivalently read as
  - $p$  if **and only** if  $q$
  - $p$  is a **necessary and sufficient** condition for  $q$
  - if  $p$  then  $q$ , and **conversely**
  - $p$  iff  $q$
- Examples
  - $x > 0$  if and only if  $x^2$  is positive
  - The alarm goes off iff a burglar breaks in
  - You may have pudding iff you eat your meat

## Exercise: Which of the following biconditionals is true?

---

- $x^2 + y^2 = 0$  if and only if  $x=0$  and  $y=0$

True. Both implications hold

- $2 + 2 = 4$  if and only if  $\sqrt{2} < 2$

True. Both implications hold.

- $x^2 \geq 0$  if and only if  $x \geq 0$

False. The implication “if  $x \geq 0$  then  $x^2 \geq 0$ ” holds.

However, the implication “if  $x^2 \geq 0$  then  $x \geq 0$ ” is false.

Consider  $x=-1$ .

The hypothesis  $(-1)^2=1 \geq 0$  but the conclusion fails.

# Converse, Inverse, Contrapositive

---

- Consider the proposition  $p \rightarrow q$ 
  - Its converse is the proposition  $q \rightarrow p$
  - Its inverse is the proposition  $\neg p \rightarrow \neg q$
  - Its contrapositive is the proposition  $\neg q \rightarrow \neg p$

# Truth Tables

---

- Truth tables are used to show/define the relationships between the truth values of
  - the individual propositions and
  - the compound propositions based on them

$p$	$q$	$p \wedge q$	$p \vee q$	$p \oplus q$	$p \Rightarrow q$	$p \Leftrightarrow q$
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	0
1	1	1	1	0	1	1

# Constructing Truth Tables

---

- Construct the truth table for the following compound proposition

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

$p$	$q$	$p \wedge q$	$\neg q$	$((p \wedge q) \vee \neg q)$
0	0	0	1	1
0	1	0	0	0
1	0	0	1	1
1	1	1	0	1

# Outline

---

- Defining Propositional Logic
  - Propositions
  - Connectives
  - Precedence of Logical Operators
  - Truth tables
- **Usefulness of Logic**
  - **Bitwise operations**
  - **Logic in Theoretical Computer Science (SAT)**
  - **Logic in Programming**
- Logical Equivalences
  - Terminology
  - Truth tables
  - Equivalence rules



# Usefulness of Logic

---

- Logic is more precise than natural language
  - You may have cake or ice cream.
    - Can I have both?
  - If you buy your air ticket in advance, it is cheaper.
    - Are there or not cheap last-minute tickets?
- For this reason, logic is used for hardware and software specification
  - Given a set of logic statements,
  - One can decide whether or not they are satisfiable (i.e., consistent), although this is a costly process...

# Bitwise Operations

---

- Computers represent information as bits (binary digits)
- A bit string is a sequence of bits
- The length of the string is the number of bits in the string
- Logical connectives can be applied to bit strings of equal length
- Example

0110 1010 1101  
0101 0010 1111

Bitwise OR 0111 1010 1111

Bitwise AND ...

Bitwise XOR ...

# Logic in TCS

---

- **What is SAT?** SAT is the problem of determining whether or not a sentence in propositional logic (PL) is satisfiable.
  - **Given:** a PL sentence
  - **Question:** Determine whether or not it is satisfiable
- Characterizing SAT as an NP-complete problem (complexity class) is at the foundation of Theoretical Computer Science.
- What is a PL sentence? What does satisfiable mean?

# Logic in TCS: A Sentence in PL

---

- A Boolean variable is a variable that can have a value 1 or 0. Thus, Boolean variable is a proposition.
- A term is a Boolean variable
- A literal is a term or its negation
- A clause is a disjunction of literals
- A sentence in PL is a conjunction of clauses
- Example:  $(a \vee b \vee \neg c \vee \neg d) \wedge (\neg b \vee c) \wedge (\neg a \vee c \vee d)$
- A sentence in PL is satisfiable iff
  - we can assign a truth value
  - to each Boolean variables
  - such that the sentence evaluates to true (i.e., holds)

# SAT in TCS

---

- Problem
  - **Given:** A sentence in PL (a complex proposition), which is
    - Boolean variables connected with logical connectives
    - Usually, as a conjunction of clauses (CNF = Conjunctive Normal Form)
  - **Question:**
    - Find an assignment of truth values [0 | 1] to the variables
    - That makes the sentence true, i.e. the sentence holds

# Logic in Programming: Example 1

---

- Say you need to define a conditional statement as follows:
  - Increment  $x$  if the following condition holds  
 $(x > 0 \text{ and } x < 10) \text{ or } x=10$
- You may try: `If (0<x<10 OR x=10) x++;`
- Can't be written in C++ or Java
- How can you modify this statement by using logical equivalence
- Answer: `If (x>0 AND x<=10) x++;`

# Logic in Programming: Example 2

---

- Say we have the following loop

```
While
```

```
((i<size AND A[i]>10) OR  
(i<size AND A[i]<0) OR  
(i<size AND (NOT (A[i]!=0 AND NOT (A[i]>=10))))))
```

- Is this a good code? Keep in mind:
  - Readability
  - Extraneous code is inefficient and poor style
  - Complicated code is more prone to errors and difficult to debug
  - Solution? Comes later...

# Outline

---

- Defining Propositional Logic
  - Propositions
  - Connectives
  - Precedence of Logical Operators
  - Truth tables
- Usefulness of Logic
  - Bitwise operations
  - Logic in Theoretical Computer Science (SAT)
  - Logic in Programming
- **Logical Equivalences**
  - **Terminology**
  - **Truth tables**
  - **Equivalence rules**



# Propositional Equivalences: Introduction

---

- In order to manipulate a set of statements (here, logical propositions) for the sake of mathematical argumentation, an important step is to replace
  - one statement with
  - another equivalent statement
  - (i.e., with the same truth value)
- Below, we discuss
  - Terminology
  - Establishing logical equivalences using truth tables
  - Establishing logical equivalences using known laws (of logical equivalences)

# Terminology:

## Tautology, Contradictions, Contingencies

- Definitions
  - A compound proposition that is always true, no matter what the truth values of the propositions that occur in it is called a tautology
  - A compound proposition that is always false is called a contradiction
  - A proposition that is neither a tautology nor a contradiction is a contingency
- Examples
  - A simple tautology is  $p \vee \neg p$
  - A simple contradiction is  $p \wedge \neg p$

# Logical Equivalences: Definition

---

- **Definition:** Propositions  $p$  and  $q$  are logically equivalent if  $p \leftrightarrow q$  is a tautology.
- Informally,  $p$  and  $q$  are equivalent if whenever  $p$  is true,  $q$  is true, and vice versa
- Notation:  $p \equiv q$  ( $p$  is equivalent to  $q$ ),  $p \leftrightarrow q$ , and  $p \Leftrightarrow q$
- Alert:  $\equiv$  is not a logical connective  $\$\\equiv\$\$

# Logical Equivalences: Example 1

---

- Are the propositions  $(p \rightarrow q)$  and  $(\neg p \vee q)$  logically equivalent?
- To find out, we construct the truth tables for each:

$p$	$q$	$p \rightarrow q$	$\neg p$	$\neg p \vee q$
0	0			
0	1			
1	0			
1	1			

The two columns in the truth table are identical, thus we conclude that

$$(p \rightarrow q) \equiv (\neg p \vee q)$$

# Logical Equivalences: Example 1

---

- Show that  $(p \rightarrow r) \vee (q \rightarrow r) \equiv (p \wedge q) \rightarrow r$  (Exercise 25 from Rosen)

$$(p \rightarrow r) \vee (q \rightarrow r) \equiv (p \wedge q) \rightarrow r$$

$p$	$q$	$r$	$p \rightarrow r$	$q \rightarrow r$	$(p \rightarrow r) \vee (q \rightarrow r)$	$p \wedge q$	$(p \wedge q) \rightarrow r$
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

# Propositional Equivalences: Introduction

---

- In order to manipulate a set of statements (here, logical propositions) for the sake of mathematical argumentation, an important step is to replace
  - one statement with
  - another equivalent statement
  - (i.e., with the same truth value)
- Below, we discuss
  - Terminology
  - **Establishing logical equivalences using truth tables**
  - **Establishing logical equivalences using known laws (of logical equivalences)**

# Logical Equivalences: Cheat Sheet

---

- Table of logical equivalences can be found in Rosen (Table 6, page 27)
- These and other can be found in a handout on the course web page:  
<http://www.cse.unl.edu/~choueiry/S12-235/files/LogicalEquivalences.pdf>
- Let's take a quick look at this Cheat Sheet

# Using Logical Equivalences: Example 1

---

- Logical equivalences can be used to construct additional logical equivalences
- Example: Show that  $(p \wedge q) \rightarrow q$  is a tautology

0.  $(p \wedge q) \rightarrow q$

1.  $\equiv \neg(p \wedge q) \vee q$

Implication Law on 0

2.  $\equiv (\neg p \vee \neg q) \vee q$

De Morgan's Law (1<sup>st</sup>) on 1

3.  $\equiv \neg p \vee (\neg q \vee q)$

Associative Law on 2

4.  $\equiv \neg p \vee 1$

Negation Law on 3

5.  $\equiv 1$

Domination Law on 4



# My Advice

---

- Remove double implication
- Replace implication by disjunction
- Push negation inwards
- Distribute

# Using Logical Equivalences: Example 2

---

- Example (Exercise 17)\*: Show that  $\neg(p \leftrightarrow q) \equiv (p \leftrightarrow \neg q)$
- Sometimes it helps to start with the second proposition ( $p \leftrightarrow \neg q$ )

0.  $(p \leftrightarrow \neg q)$

1.  $\equiv (p \rightarrow \neg q) \wedge (\neg q \rightarrow p)$

2.  $\equiv (\neg p \vee \neg q) \wedge (q \vee p)$

3.  $\equiv \neg(\neg((\neg p \vee \neg q) \wedge (q \vee p)))$

2

4.  $\equiv \neg(\neg(\neg p \vee \neg q) \vee \neg(q \vee p))$

Law...

5.  $\equiv \neg((p \wedge q) \vee (\neg q \wedge \neg p))$

Law

6.  $\equiv \neg((p \vee \neg q) \wedge (p \vee \neg p) \wedge (q \vee \neg q) \wedge (q \vee \neg p))$

7.  $\equiv \neg((p \vee \neg q) \wedge (q \vee \neg p))$

8.  $\equiv \neg((q \rightarrow p) \wedge (p \rightarrow q))$

9.  $\equiv \neg(p \leftrightarrow q)$

Equivalence Law on 0

Implication Law on 1

Double negation on

De Morgan's

De Morgan's

Distribution Law

Identity Law

Implication Law

Equivalence Law

\*See Table 8 (p 25) but you are not allowed to use the table for the proof

# Using Logical Equivalences: Example 3

---

- Show that  $\neg(q \rightarrow p) \vee (p \wedge q) \equiv q$ 
  0.  $\neg(q \rightarrow p) \vee (p \wedge q)$
  1.  $\equiv \neg(\neg q \vee p) \vee (p \wedge q)$  Implication Law
  2.  $\equiv (q \wedge \neg p) \vee (p \wedge q)$  De Morgan's & Double negation
  3.  $\equiv (q \wedge \neg p) \vee (q \wedge p)$  Commutative Law
  4.  $\equiv q \wedge (\neg p \vee p)$  Distributive Law
  5.  $\equiv q \wedge 1$  Identity Law
  - $\equiv q$  Identity Law

# Proving Logical Equivalences: Summary

---

- Proving two PL sentences  $A, B$  are equivalent using **TT** + **EL**
  1. Verify that the 2 columns of  $A, B$  in the truth table are the same (i.e.,  $A, B$  have the same models)
  2. Verify that the column of  $(A \rightarrow B \wedge B \rightarrow A)$  in the truth table has all-1 entries (it is a tautology)
  3. Apply a sequence of Equivalence Laws
    - Put  $A, B$  in CNF, they should be the same
    - Sequence of equivalence laws: Biconditional, implication, moving negation inwards, distributivity
  4. Apply a sequence of inference laws
    - Starting from one sentence, usually the most complex one,
    - Until reaching the second sentence

# Logic in Programming: Example 2 (revisited)

---

- Recall the loop

```
While  
  ((i<size AND A[i]>10) OR  
   (i<size AND A[i]<0) OR  
   (i<size AND (NOT (A[i]!=0 AND NOT (A[i]>=10))))))
```

- Now, using logical equivalences, simplify it!
- Using De Morgan's Law and Distributivity

```
While ((i<size) AND  
       ((A[i]>10 OR A[i]<0) OR  
        (A[i]==0 OR A[i]>=10)))
```

- Noticing the ranges of the 4 conditions of  $A[i]$

```
While ((i<size) AND (A[i]>=10 OR A[i]<=0))
```

# Programming Pitfall Note

---

- In C, C++ and Java, applying the commutative law is not such a good idea.
- For example, consider accessing an integer array  $A$  of size  $n$ :

```
if (i < n && A[i] == 0) i++;
```

is not equivalent to

```
if (A[i] == 0 && i < n) i++;
```