The Foundations: Logic and Proofs

Chapter 1, Part I: Propositional Logic

ICSI 210 Discrete Structures

Qi Wang

Department of Computer Science

University at Albany
State University of New York

Chapter Summary

- Propositional Logic
  - The Language of Propositions
  - Applications
  - Logical Equivalences
- Predicate Logic
  - The Language of Quantifiers

Propositional Logic

Section 1.1

Propositions

- A *proposition* is a declarative sentence that is either true or false.
- Examples of propositions:
  a) Washington, D.C., is the capital of the United States of America.
  b) Toronto is the capital of Canada.
  c) $1 + 0 = 1$
  d) $0 + 0 = 2$
- Examples that are not propositions.
  a) Sit down!
  b) What time is it?
  c) $x + 1 = 2$
  d) $x + y = z$

Propositional Logic

- Constructing Propositions
  - **Propositional Variables**: *p*, *q*, *r*, *s*, …
  - The proposition that is always true is denoted by **T** and the proposition that is always false is denoted by **F**.
  - **Compound Propositions**; constructed from logical connectives and other propositions
    - Negation $\neg$
    - Conjunction $\land$
    - Disjunction $\lor$
    - Implication $\rightarrow$
    - Biconditional $\leftrightarrow$

Compound Propositions: Negation

- The *negation* of a proposition p is denoted by $\neg p$ and has this truth table:

p	**¬p**
T	F
F	T

- **Example**:

  $\boldsymbol{p}$: The earth is round.

  $\neg \boldsymbol{p}$: The earth is *not* round.
  It is not the case that the earth is round.

Conjunction

- The *conjunction* of propositions p and q is denoted by $p \land q$ and has this truth table:

p	**q**	**p ∧ q**
T	T	**T**
T	F	F
F	T	F
F	F	F

- **Example**:

  p : I am at home. q : It is raining.
  $p \land q$: I am at home and it is raining.

Disjunction (Inclusive Or)

- The *disjunction* of propositions p and q is denoted by $p \vee q$ and has this truth table:

p	q	p ∨ q
T	**T**	**T**
T	F	T
F	**T**	T
F	F	F

"Inclusive disjunction" or "Inclusive Or" since both *p* and *q* can be true.

- **Example**:

p : I am at home. q : It is raining.

$p \vee q$: I am at home **or** it is raining.

It is possible that both "I am at home." and "it is raining. " are true.

Exclusive Or (Xor)

- In English, *or* can be *inclusive* or *exclusive*.
- The *Exclusive Or* of propositions p and q is denoted by $p \oplus q$ and has this truth table:

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

$\}$ *Exclusive* means *p* and *q* have different values.

Exclusive Or (Xor)

- “Soup **or** salad comes with this entrée.”
  - Get soup or salad, but not both. (Get one choice only.)

Implication

- $p \to q$ is a *conditional statement* or *implication* which is read as "if p, then q" and has this truth table:

p	q	p →q
T	T	T
T	F	F
F	T	T
F	F	T

- **Example**:

  $\boldsymbol{p}$ (hypothesis): I am at home.
  $\boldsymbol{q}$ (*conclusion*): It is raining.
  $\boldsymbol{p \to q}$: If I am at home then it is raining.

Implication

- An obligation or contract.

  $\text{“}\overbrace{\text{If you get 100\% on the final}}^{p}\text{, then }\overbrace{\text{you will get an A}}^{q}\text{.”}$

  - The professor is obligated to assign you a grade of A if you get 100% on the final.

p	q	p →q	
T	**T**	**T**	**You get 100% on the final, then you will get an A.”**
T	F	F	You get 100% on the final, you will not get an A. - **Violation!**
F	T	**T**	You didn’t get 100% on the final … In logic, the whole statement is treated **true** since it wasn’t proved false.
F	F	**T**	

Converse, Contrapositive, and Inverse

- From $p \to q$ we can form new conditional statements .
  - $q \to p$ is the **converse** of $p \to q$ (SWAP)
  - $\neg q \to \neg p$ is the **contrapositive** of $p \to q$ $(\text{SWAP} + \text{NEGATE})$
  - $\neg p \to \neg q$ is the **inverse** of $p \to q$ (NEGATE)

- **Example**:

p: It's raining. ***q***: I am not going to town.

Converse ($q \to p$**)**: If I do not go to town, then it is raining.

Contrapositive ($\neg q \to \neg p$**)**: If I go to town, then it is not raining.

Inverse ($\neg p \to \neg q$**)**: If it is not raining, then I will go to town.

Biconditional

- $p \leftrightarrow q$, read as "p **if and only if** q.", is a *biconditional* proposition this truth table:

p	**q**	**p ↔ q**
T	**T**	**T**
T	F	F
F	T	F
F	**F**	**T**

- It is true **when p and q have the same truth values** and is false otherwise.

Biconditional

- Biconditional statements are also called bi-implications.
  - $p \leftrightarrow q$: $p \rightarrow q$ and $p \rightarrow q$
- Example:

  $\boldsymbol{p}$: I am at home. $\boldsymbol{q}$:It is raining.

  $\boldsymbol{p} \leftrightarrow \boldsymbol{q}$: I am at home *if and only if* it is raining.

Precedence of Logical Operators

Operator	Precedence
$\neg$	1
$\wedge$ $\vee$	2 3
$\rightarrow$ $\leftrightarrow$	4 5

- **Example:**

  $p \vee q \rightarrow \neg r$:
  - The order of operations is : $\neg$, $\vee$, and then $\rightarrow$.

  $p \vee (q \rightarrow \neg r)$:
  - The order of operations is : $\neg$, $\rightarrow$, and then $\vee$.
  - **Always do Parentheses first.**

Truth Tables For Compound Propositions

- Construction of a truth table:
- Rows
  - Need a row for every possible combination of values for the atomic propositions.
- Columns
  - Need a column for the compound proposition (usually at far right)
  - Need a column for the truth value of each expression(operation) that occurs in the compound proposition as it is built up.

Truth Tables For Compound Propositions

- How many rows are there in a truth table with *n* propositional variables?
    - Each propositional variable's value can be T or F, therefore, there are 2^n rows in a truth table with n variables.

- Note that this means that with *n* propositional variables, we can construct 2^n distinct (i.e., not equivalent) propositions.

Truth Tables For Compound Propositions

- The truth table of $p \vee q \rightarrow \neg r$: 8($2^3$) rows and 6 columns.

p	**q**	**r**	**¬r**	**p ∨ q**	**p ∨ q → ¬r**

Truth Tables For Compound Propositions $p \vee q \rightarrow \neg r$

- Add a column for each propositional variable(*p,q, and r*) in the same order as they are in the proposition.

p	q	r	$\neg$r	p $\vee$ q	p $\vee$ q $\rightarrow$ $\neg$r

Truth Tables For Compound Propositions $p \vee q \to \neg r$

- Add a column for each operator in the order of operations (¬, ∨, and then →).

p	q	r	¬r	p ∨ q	p ∨ q → ¬r

- The far-right column is for the proposition.

Truth Tables For Compound Propositions

- How to fill up the rows – the combinations of values for the atomic propositions.
  - Start in the left-most column, divide it evenly, fill the first half with T’s and the second half with F’s.
  - Then move right to the next column, repeat the previous step in the first half of the column and the second half of the column. Repeat until the last column is filled with alternating T's and F's.
  - The first row will be all **T**'s.
  - The last row will be all **F**'s.

Truth Tables For Compound Propositions $p \vee q \rightarrow \neg r$

- Start in the left-most column, divide it evenly, fill the first half with T's and the second half with F's.

p	q	r	¬r	p ∨ q	p ∨ q → ¬r
T					
T					
T					
T					
F					
F					
F					
F					

Truth Tables For Compound Propositions $p \vee q \rightarrow \neg r$

- Then move right to the next column, repeat the previous step in the first half of the column and the second half of the column until the last column is filled with alternating T's and F's.

p	q	r	¬r	p ∨ q	p ∨ q → ¬r
T	T				
T	T				
T	F				
T	F				
F	T				
F	T				
F	F				
F	F				

Truth Tables For Compound Propositions $p \vee q \rightarrow \neg r$

- Then move right to the next column, repeat the previous step in the first half of the column and the second half of the column until the last column is filled with **alternating** T's and F's.

p	**q**	**r**	**¬r**	**p ∨ q**	**p ∨ q → ¬r**
T	T	T			
T	T	F			
T	F	T			
T	F	F			
F	T	T			
F	T	F			
F	F	T			
F	F	F			

Truth Tables For Compound Propositions $p \vee q \rightarrow \neg r$

- The first row will be all T's.
- The last row will be all F's.

p	q	r	¬r	p ∨ q	p ∨ q → ¬r
T	T	T			
T	T	F			
T	F	T			
T	F	F			
F	T	T			
F	T	F			
F	F	T			
F	F	F			

Truth Tables For Compound Propositions $p \vee q \rightarrow \neg r$

- Continue to fill the rest of columns with the corresponding truth values.

p	**q**	**r**	**¬r**	**p ∨ q**	**p ∨ q → ¬r**
T	T	T	F	T	F
T	T	F	T	T	T
T	F	T	F	T	F
T	F	F	T	T	T
F	T	T	F	T	F
F	T	F	T	T	T
F	F	T	F	F	T
F	F	F	T	F	T

Equivalent Propositions

- Two propositions are **e***quivalent* if they always have the same truth value.
- For example, a conditional (p →q) is equivalent to its contrapositive (¬q → ¬ p).

p	**q**	**¬ p**	**¬ q**	**p →q**	**¬q → ¬ p**
T	T	F	F	T	T
T	F	F	T	F	F
F	T	T	F	T	T
F	F	T	T	T	T

Non-Equivalence

- Two propositions are not **e***quivalent* if they don't always have the same truth value.
- For example, neither the converse nor inverse of an implication (p →q) is not equivalent to the implication.

p	**q**	**¬ p**	**¬ q**	**p →q**	**¬ p →¬ q**	**q → p**
					Inverse	Converse
T	T	F	F	T	T	T
T	F	F	T	**F**	**T**	**T**
F	T	T	F	T	F	F
F	F	T	T	T	T	T

Logic and Bit Operations

- Computers represent information using bits.
- A bit is a symbol with two possible values, namely, 0 (zero) and 1 (one).
  - 0 represents false (F).
  - 1 represents true (T).

Truth Value	**Bit**
T	1
F	0

- A **Boolean variable's value is** either true or false.
- A Boolean variable can be represented using a bit.

Logic and Bit Operations

- Computer **bit operations** correspond to the logical connectives.

Bitwise OR (∨, Disjunction, Inclusive OR)
Bitwise AND (∧, Conjunction)
Bitwise XOR ($\bigoplus$ Disjunction, *Exclusive OR)*

Truth Tables for Bit Operations

- How to fill up the rows – the combinations of values for the atomic propositions.
  - Start in the left-most column, divide it evenly, fill the first half with **0’s** and the second half with **1’s**.
  - Then move right to the next column, repeat the previous step in the first half of the column and the second half of the column. Repeat until the last column is filled with alternating **0's** and **1's**.
  - The first row will be all 0's.
  - The last row will be all **1's**.

Logic and Bit Operations

- In the truth tables for the operators ∧, ∨, and ⊕, replace T by a 1 bit and F by a 0 bit.

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

- **The first row will be all 0’s.**
- **The last row will be all 1’s.**

Logic and Bit Operations

- A *bit string* is a sequence of zero or more bits.
- The *length* of this string is the number of bits in the string.

Logic and Bit Operations

- **Example:** Find the bitwise *OR*, bitwise *AND*, and bitwise *XOR* of the bit strings 01 1011 0110 and 11 0001 1101.

Bitwise OR	Bitwise AND	Bitwise XOR
01 1011 0110	01 1011 0110	01 1011 0110
$\underline{11\ 0001\ 1101}$	$\underline{11\ 0001\ 1101}$	$\underline{11\ 0001\ 1101}$
11 1011 1111	01 0001 0100	10 1010 1011

Applications of Propositional Logic

Section 1.2

Applications of Propositional Logic

- Translating English to Propositional Logic
- Boolean Searches
- Logic Circuits

Translating English Sentences

- Steps to convert an English sentence to a statement in propositional logic:
  - Identify atomic propositions and represent using propositional variables.
  - Determine appropriate logical connectives

"If I go to Harry's **or** to the country, I will **not** go shopping."

***p*:** I go to Harry's.

q: I go to the country.

***r*:** I will go shopping.

$(p \vee q) \rightarrow \neg r$: If *p* or *q* then not *r*.

Translating English Sentences

- Another example:

  "You **cannot** ride the roller coaster **if** you are under 4 feet tall **unless** you are older than 16 years old."

  q: You can ride the roller coaster.
  r: You are under 4 feet tall.
  s: You are older than 16 years old.

  $(r \wedge \neg s) \rightarrow \neg q$: If *r* and *not* *s* then not *q*.

- Is there another solution?

Boolean Searches

- Logical connectives are used extensively in searches of large collections of information, such as indexes of Web pages.

Connective	
AND	match records that contain **both** of two search terms
OR	match **one or both** of two search terms
NOT	**exclude** a particular search term. (sometimes written as AND NOT)

Boolean Searches

- **Web Page Searching** Most Web search engines support Boolean searching techniques, which usually can help find Web pages about particular subjects.
- For example,

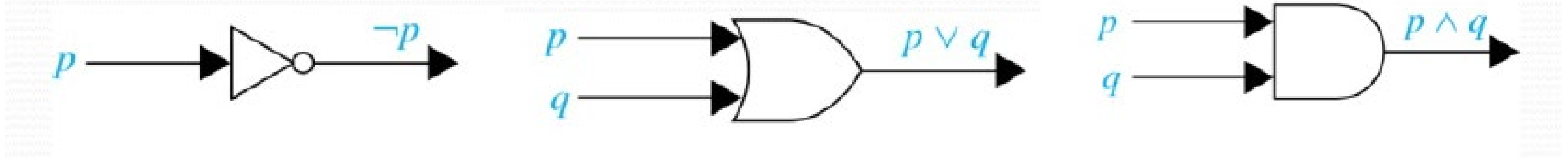
To find Web pages about **universities in New Mexico**

$\downarrow$

NEW ***AND*** MEXICO ***AND*** UNIVERSITIES

Logic Circuits

- In electronic circuits, each input/output signal can be viewed as a 0 or 1.
  - 0 represents **False.**
  - 1 represents **True.**
- Complicated circuits are constructed from three basic circuits called gates:, **OR** gate and **AND** gate.



NOT gate/ Inverter

OR gate

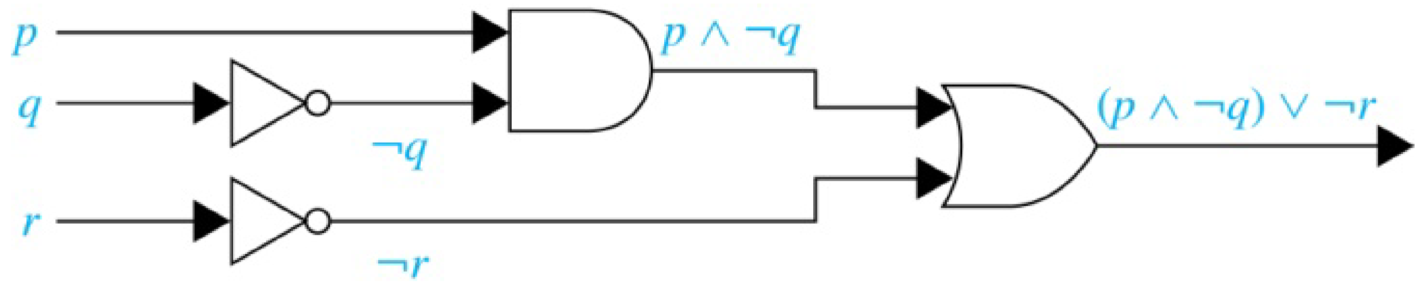
AND gate

Logic Circuits

NOT gate/ Inverter	takes an input bit and produces the **negation** of that bit.
OR gate	takes two input bits and produces the value equivalent to the **disjunction** of the two bits.
AND gate	takes two input bits and produces the value equivalent to the **conjunction** of the two bits.

Logic Circuits

- More complicated digital circuits can be constructed by combining these basic circuits to produce the desired output (E.g., $(p \wedge \neg q) \vee \neg r$) given the input signals.
    - Build a circuit for each piece of the output expression and then combining them.



Example

Problem: Build a digital circuit that produces the output

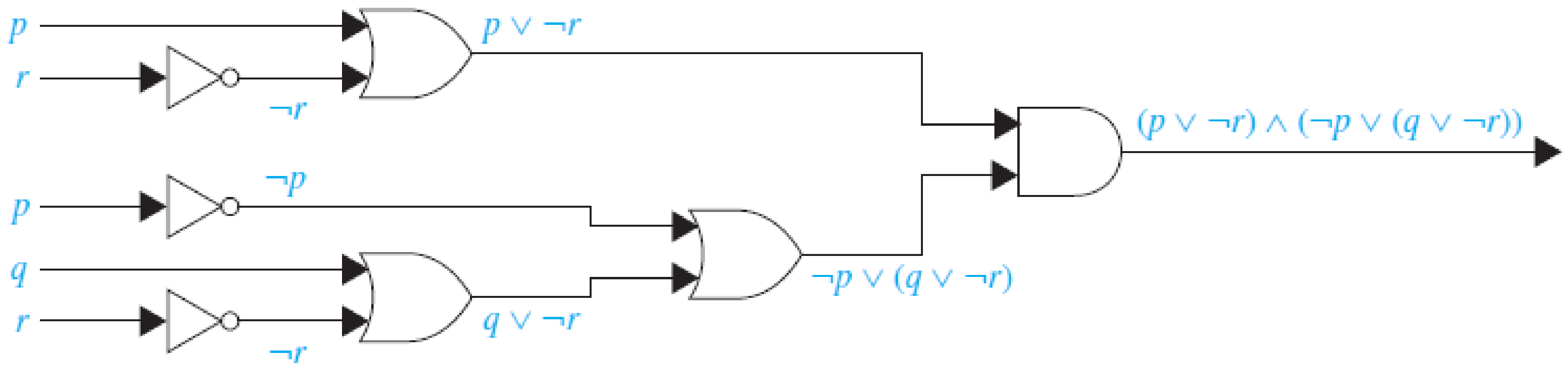
$$(p \vee \neg r) \wedge (\neg p \vee (q \vee \neg r))$$

when given input signals *p, q*, and r.

Example

Solution:

To construct the desired circuit, we build separate circuits for the corresponding bit operations and combine them using the appropriate gates.



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

Propositional Equivalences

Section 1.3

Propositional Equivalences

- Tautologies, Contradictions, and Contingencies.
- Logical Equivalence
    - Important Logical Equivalences
    - Showing Logical Equivalence

Tautologies, Contradictions, and Contingencies

- A ***tautology*** is a proposition which is always **true**.
  - Example: $p \vee \neg p$
- A ***contradiction*** is a proposition which is always **false**.
  - Example: $p \wedge \neg p$
- A ***contingency*** is a proposition which is **neither a tautology nor a contradiction**, such as p

P	**¬p**	**p ∨¬p**	**p ∧¬p**
T	F	T	F
F	T	T	F

Logically Equivalent

- Two compound propositions *p* and *q* are logically equivalent if $p \leftrightarrow q$ is a tautology (need proofs).
- Two compound propositions are logically equivalent if and only if the columns in a truth table giving their truth values agree.
  - $\neg p \vee q$ is equivalent to $p \rightarrow q$

p	**q**	**¬p**	**¬p ∨ q**	**p→ q**
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

De Morgan’s Laws

$$\neg(p \wedge q) \equiv \neg p \vee \neg q$$ First Law

$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$ Second Law

- This truth table shows that De Morgan’s Second Law holds.

p	**q**	**¬p**	**¬q**	**(p∨q)**	**¬(p∨q)**	**¬p∧¬q**
T	T	F	F	T	F	F
T	F	F	T	T	F	F
F	T	T	F	T	F	F
F	F	T	T	F	T	T

Key Logical Equivalences

- Identity Laws: $p \wedge T \equiv p$ $\quad p \vee F \equiv p$
- Domination Laws: $p \vee T \equiv T$ $\quad p \wedge F \equiv F$
- Idempotent laws: $p \vee p \equiv p$ $\quad p \wedge p \equiv p$
- Double Negation Law: $\neg(\neg p) \equiv p$
- Negation Laws: $p \vee \neg p \equiv T$ $\quad p \wedge \neg p \equiv F$

Key Logical Equivalences (*cont*)

- Commutative Laws: $p \vee q \equiv q \vee p \qquad p \wedge q \equiv q \wedge p$

- Associative Laws: $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
  $(p \vee q) \vee r \equiv p \vee (q \vee r)$

- Distributive Laws: $(p \vee (q \wedge r)) \equiv (p \vee q) \wedge (p \vee r)$
  $(p \wedge (q \vee r)) \equiv (p \wedge q) \vee (p \wedge r)$

- Absorption Laws: $p \vee (p \wedge q) \equiv p \quad p \wedge (p \vee q) \equiv p$

Logical Equivalences Involving Conditional Statements and Biconditional Statements

$p \to q \equiv \neg p \vee q$

$p \to q \equiv \neg q \to \neg p$

$p \vee q \equiv \neg p \to q$

$p \wedge q \equiv \neg(p \to \neg q)$

$\neg(p \to q) \equiv p \wedge \neg q$

$(p \to q) \wedge (p \to r) \equiv p \to (q \wedge r)$

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

$(p \to q) \vee (p \to r) \equiv p \to (q \vee r)$

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

$p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$

$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$

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

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

Logical Equivalences

- Given two expressions *A* and *B*, to prove that $A \equiv B$ we produce a series of equivalences beginning with A and ending with B.

$$\begin{aligned} A &\equiv A_1 \\ &\vdots \\ A_n &\equiv B \end{aligned}$$

Equivalence Proofs

Example: Show that $\neg(p \vee (\neg p \wedge q)) \equiv \neg p \wedge \neg q$.

Solution:

$$\begin{array}{lcll}
\neg(p \vee (\neg p \wedge q)) & \equiv & \neg p \wedge \neg(\neg p \wedge q) & \text{by the second De Morgan law} \\
 & \equiv & \neg p \wedge [\neg(\neg p) \vee \neg q] & \text{by the first De Morgan law} \\
 & \equiv & \neg p \wedge (p \vee \neg q) & \text{by the double negation law} \\
 & \equiv & (\neg p \wedge p) \vee (\neg p \wedge \neg q) & \text{by the second distributive law} \\
 & \equiv & F \vee (\neg p \wedge \neg q) & \text{because } \neg p \wedge p \equiv F \\
 & \equiv & (\neg p \wedge \neg q) \vee F & \text{by the commutative law} \\
 & & & \text{for disjunction} \\
 & \equiv & (\neg p \wedge \neg q) & \text{by the identity law for } \mathbf{F}
\end{array}$$

Equivalence Proofs

Example: Show that $(p \wedge q) \rightarrow (p \vee q)$ is a tautology.

Solution:

Logical Equivalences Involving Conditional Statements

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

$(p \wedge q) \rightarrow (p \vee q) \equiv \neg(p \wedge q) \vee (p \vee q)$	by this logical equivalence
$\equiv (\neg p \vee \neg q) \vee (p \vee q)$	by the first De Morgan law
$\equiv (\neg p \vee p) \vee (\neg q \vee q)$	by the associative and commutative laws for disjunction
$\equiv \mathbf{T} \vee \mathbf{T}$	by Example 1 and the commutative law for disjunction
$\equiv \mathbf{T}$	by the domination law

The Foundations: Logic and Proofs

Chapter 1, Part II: Predicate Logic

ICSI 210 Discrete Structures

Predicates and Quantifiers

Section 1.4

Propositional Logic

- Propositional logic cannot adequately express the meaning of all statements in mathematics and in natural language.

  *“**Every computer** connected to the university network is functioning properly.”*

  All computers are ...

- No rules of propositional logic allow us to conclude the truth of the statement

  *“**MATH3** is functioning properly,”*

  There exists one computer such as ..
  Or
  There exists at least one computer such as

Predicate Logic

- **Predicate logic** talks about objects, their properties, and their relations.
- **Predicate logic** can be used to express the meaning of a wide range of statements in mathematics and computer science.

Predicate Logic

- Predicate logic uses the following new features:
  - Variables: *x*, *y*, *z*
  - Predicates: ***P***(*x*), ***M***(*x*)
  - Quantifiers (*to be covered in a few slides*):
- ***Propositional functions*** **(**a generalization of propositions) contain variables and a predicate, e.g., *P*(*x*).
  - Variables can be replaced by elements from their domain.

Predicate Logic

- Statements involving variables:
  - $x > 3$
  - $x = y + 3$
  - $x + y = z$
  - Computer *x* is under attack by an intruder.
  - Computer *x* is functioning properly.
- They are often found in mathematical assertions, in computer programs, and in system specifications.

Predicate Logic

- "$x > 3$" has two parts.

$$\text{“}\underbrace{\underline{x}}_{\text{subject}}\ \underbrace{\underline{\textbf{is greater than 3}}}_{\text{predicate}}\text{”}$$

$P(x)$

  - The 1st part: the variable ***x***, is the **subject** of the statement.
  - The 2nd part: **"is greater than 3",** the **predicate**, which refers to a property that the subject of the statement can have.

Predicate Logic

- *P(x): x* is greater than 3.
- *P(x):* the value of the **propositional function** *P* at *x*.
- Once a value has been assigned to the variable *x*, *P(x)* becomes a proposition and has a truth value.

Propositional Functions

- For example, let *P(x)* denote "$x > 0$" and the domain *U* be the integers.

  P(-3) is false.

  P(0) is false.

  P(3) is true.

Propositional Functions

- For example, let *Q(x, y)* denote the statement "$x = y + 3$." What are the truth values of the propositions *Q(*1, 2*)* and *Q(*3, 0*)*?

  *Q(*1, 2*)* is "1 = 2 + 3," which is false.
  *Q(*3, 0*)* is "3 = 0 + 3," which is true.

Propositional Functions

- Let "$x + y = z$" be denoted by $R(x, y, z)$ and U (for all three variables) be the integers. Find these truth values:

  R$(2,-1,5)$: F

  R$(3,4,7)$: T

  R$(x, 3, z)$: Not a Proposition

Propositional Functions

- Now let "$x - y = z$" be denoted by $Q(x, y, z)$, with U as the integers. Find these truth values:

  $Q(2,-1,3)$: T

  $Q(3,4,7)$: F

  $Q(x, 3, z)$: Not a Proposition

Propositional Functions

- Propositional functions occur in computer programs.
- Consider the statement

  if $x > 0$ **then** $x := x + 1$.

  $P(x) – x > 0$

  When $P(x)$ is true, variable x is increased by 1.

Propositional Functions

- Predicates are also used to establish the correctness of computer programs, that is, to show that computer programs always produce the desired output when given valid input.
- **Preconditions:** the statements that describe valid input.
- **Postconditions:** the conditions that the output should satisfy when the program has run.

Propositional Functions

- For example (swap x and y),

```
temp := x
x := y
y := temp
```

- **Precondition:**

P(x, y), where *P(x, y)* is the statement "*x* = *a* and *y* = *b*," where *a* and *b* are the values of *x* and *y* before we run the program.

- **Postcondition:**

Q(x, y), where *Q(x, y)* is the statement "*x* = *b* and *y* = *a*."

Quantifiers

- Quantification expresses the extent to which **a predicate is true over a range of elements.**
- We need *quantifiers* to express the meaning of English words including ***all*** and ***some***:
  - "All men are Mortal."
  - "Some cats do not have fur."
  - "All computers are functioning properly."
  - "Some computers are functioning properly."

Quantifiers

- The two most important quantifiers are:
  - *Universal Quantifier,* **“**For all,”   symbol: ∀
  - *Existential Quantifier*, “There exists,” symbol: ∃

- $\forall x\ P(x)$ asserts $P(x)$ is true for <u>every</u> *x* in the *domain*.
  - Read as “For all x, P(x)” or “For every x, P(x).”
  - An element for which *P(x)* is false is called a **counterexample** of $\forall x P(x)$.

Quantifiers

- $\exists x\ P(x)$ asserts $P(x)$ is true for <u>some</u> x in the *domain*.
  - Read as "For some x, P(x)", or as "There is an x such that P(x)," or "For at least one x, P(x)."

Universal Quantifier

Examples:

1. If *P(x)* denotes "$x > 0$" and U is the integers, then $\forall x\, P(x)$ is false. P(0) is false(a counterexample).

2. If P(x) denotes "$x > 0$" and U is the positive integers, then $\forall$x P(x) is true.

3. If *P(x)* denotes "*x* is even" and U is the integers, then

   $\forall x P(x)$ is false. P(1) is false (a counterexample).

Existential Quantifier

Examples:

1. If *P(x)* denotes "$x > 0$" and U is the integers, then $\exists x\, P(x)$ is true. It is also true if U is the positive integers.

2. If *P(x)* denotes "$x < 0$" and U is the positive integers, then $\exists x\, P(x)$ is false.

3. If *P(x)* denotes "*x* is even" and U is the integers, then $\exists x\, P(x)$ is true.

Quantifier with Restricted Domains

- What do the statements mean, where the domain in each case consists of the real numbers?

Statement	**states**
$\forall \mathbf{x < 0}\ (x^2 > 0)$	that for every real number x with $x < 0$, $x^2 > 0$.
$\forall \mathbf{y \neq 0}\ (y^3 \neq 0)$	for every real number y with $y \neq 0$, we have $y^3 \neq 0$.
$\exists \mathbf{z > 0}\ (z^2 = 2)$	that there exists a real number z with $z > 0$ such that $z^2 = 2$.

Thinking about Quantifiers

- When the domain is finite, we can think of quantification as looping through the elements of the domain.
- To evaluate $\forall x\, P(x)$ loop through all x in the domain.
  - If at every step P(x) is true, then $\forall x\, P(x)$ is true.
  - If at a step P(x) is false, then $\forall x\, P(x)$ is false, and the loop terminates.
- To evaluate $\exists x\, P(x)$ loop through all x in the domain.
  - If at some step, P(x) is true, then $\exists x\, P(x)$ is true, and the loop terminates.
  - If the loop ends without finding an x for which P(x) is true, then $\exists x\, P(x)$ is false.

Properties of Quantifiers

- The truth value of $\exists x\, P(x)$ and $\forall x\, P(x)$ depend on both the propositional function *P(x)* and the domain *U*.
- **Examples**:
  1. If *U* is the positive integers and *P(x)* is the statement "$x < 2$", then $\exists x\, P(x)$ is true, but $\forall x\, P(x)$ is false.
  2. If *U* is the negative integers and *P(x)* is the statement "$x < 2$", then both $\exists x\, P(x)$ and $\forall x\, P(x)$ are true.
  3. If *U* consists of 3, 4, and 5, and *P(x)* is the statement "$x > 2$", then both $\exists x\, P(x)$ and $\forall x\, P(x)$ are true. But if *P(x)* is the statement "$x < 2$", then both $\exists x\, P(x)$ and $\forall x\, P(x)$ are false.

Precedence of Quantifiers

- The quantifiers $\forall$ and $\exists$ have **higher** precedence than all the logical operators.
- For example, $\forall x\, P(x) \vee Q(x)$ means $\boldsymbol{(\forall x\, P(x)) \vee Q(x)}$
- $\forall x\, (P(x) \vee Q(x))$ means something different.