

2008

Bryn Mawr  
College

Bethany Azuma

**[MATH 231]**

Discrete Mathematics



# Table of Contents

---

<b>Chapter 1 The Logic of Compound Statements</b> .....	<b>5</b>
1.1 Logical Form and Logical Equivalence.....	5
1.2 Conditional Statements .....	7
1.3 Valid and Invalid Arguments .....	9
1.4 Application: Digital Logic Circuits .....	11
1.5 Application: Number Systems and Circuits for Addition .....	12
<b>Chapter 2 The Logic of Quantified Statements</b> .....	<b>16</b>
2.1 Introduction to Predicates and Quantified Statements I.....	16
2.2 Introduction to Predicates and Quantified Statements II .....	17
2.3 Statements Containing Multiple Quantifiers.....	18
<b>Chapter 3 Elementary Number Theory and Methods of Proof</b> .....	<b>19</b>
3.1 Direct Proof and Counterexample I: Introduction.....	19
3.2 Direct Proof and Counterexample II: Rational Numbers .....	20
3.3 Direct Proof and Counter Example III: Divisibility.....	20
3.4 Direct Proof and Counterexample IV: Division into Cases and the Quotient-Remainder Theorem.....	21
3.5 Direct Proof and Counterexample V: Floor and Ceiling.....	21
3.6: Indirect Argument: Contradiction and Contraposition .....	21
3.7 Two Classical Theorems.....	24
3.8 Application: Algorithms .....	25
<b>Chapter 4 Sequences and Mathematical Induction</b> .....	<b>29</b>
4.1 Sequences.....	29
4.2 Mathematical Induction I.....	30
4.3 Mathematical Induction II .....	31
4.4 Strong Mathematical Induction and the Well-Ordering Principle.....	31
4.5 Application: Correctness of Algorithms.....	32
<b>Chapter 5 Set Theory</b> .....	<b>34</b>
5.1 Basic Definitions of a Set Theory .....	34
5.2 Properties of Sets.....	37
5.3 Disproofs, Algebraic Proofs, and Boolean Algebras.....	39
5.4 Russell’s Paradox and the Halting Problem.....	40
<b>Chapter 6 Counting and Probability</b> .....	<b>41</b>
6.1 Introduction.....	41
6.2 Possibility Trees and the Multiplication Rule .....	41
6.3 Counting Elements of Disjoint Sets: The Addition Rule.....	42
6.4 Counting Subsets of a Set: Combinations.....	43
6.5 r-Combinations with Repetition Allowed.....	43
6.6 The Algebra of Combinations .....	44
6.8 Probability Axioms and Expected Value .....	44
6.9 Conditional Probability, Bayer’s Formula, and Independent Events .....	44
<b>Chapter 7 Functions</b> .....	<b>46</b>
7.1 Functions Defined on General Sets .....	46

7.2 One-to-One and Onto, Inverse Functions.....	47
7.3 Application: The Pigeonhole Principle.....	48
7.4 Composition of Functions.....	48
<b>Chapter 8 Recursion .....</b>	<b>50</b>
8.1 Recursively Defined Sequences.....	50
8.2 Solving Recurrence Relations by Iteration.....	50

# Chapter 1 The Logic of Compound Statements

---

## 1.1 Logical Form and Logical Equivalence

### Statements

**Definition** A **statement** (or **proposition**) is a sentence that is true or false but not both.

### Compound Statements

Symbols:

- The symbol  $\sim$  denotes *not*
- The symbol  $\wedge$  denotes *and*
- The symbol  $\vee$  denotes *or*

Given a statement  $p$  the **negation of  $p$**  is  $\sim p$

Given the statements  $p$  and  $q$  the **conjunction of  $p$  and  $q$**  is  $p \wedge q$

Given the statements  $p$  and  $q$  the **disjunction of  $p$  and  $q$**  is  $p \vee q$

In expressions that include the symbol  $\sim$  as well as  $\wedge$  or  $\vee$ ,  $\sim$  is performed first unless there are parentheses which always come first

### Truth Values

Truth values – either true or false

**Definition** If  $p$  is a statement variable, the **negation of  $p$**  is “not  $p$ ” or “it is not the case that  $p$ ” and is denoted  $\sim p$ . It has opposite truth value from  $p$ : if  $p$  is true,  $\sim p$  is false; if  $p$  is false,  $\sim p$  is true.

**Definition** If  $p$  and  $q$  are statement variables, the **conjunction of  $p$  and  $q$**  is “ $p$  and  $q$ ,” denoted  $p \wedge q$ . It is true when, and only when, both  $p$  and  $q$  are true. If either  $p$  or  $q$  is false, or if both are false,  $p \wedge q$  is false

**Definition** If  $p$  and  $q$  are statement variables, the **disjunction of  $p$  and  $q$**  is “ $p$  or  $q$ ” denoted  $p \vee q$ . It is true when either  $p$  is true or  $q$  is true, or both  $p$  and  $q$  are true; it is false only when both  $p$  and  $q$  are false

### Evaluating the Truth of More General Compound Statements

**Definition** A **statement form** (or **propositional form**) is an expression made up of statement variables (such as  $p$ ,  $q$ , and  $r$ ) and logical connectives (such as  $\sim$ ,  $\wedge$ , and  $\vee$ ) that becomes a statement when actual statements are substituted for the component statement variables. The **truth table** for a given statement form displays the truth values that correspond to all possible combinations of truth values for its component statement variables.

### Logical Equivalence

**Definition** Two statements are called **logically equivalent** if, and only if, they have identical truth values for each possible substitution of statements for the statement variables. The logical equivalence of statement forms  $P$  and  $Q$  is denoted by writing  $P \equiv Q$ . Two statements are called **logically equivalent** if, and only if, they have logically equivalent forms when identical component statement variables are used to replace identical component statements.

### Testing Whether Two Statement Forms $P$ and $Q$ are Logically Equivalent

- Construct a truth table with one column for the truth values of  $P$  and another column for the truth values of  $Q$
- Check each combination of truth values of the statement variables to see whether the truth value of  $P$  is the same as the truth value of  $Q$ 
  - If in each row the truth value of  $P$  is the same as the truth value of  $Q$ , then  $P$  and  $Q$  are logically equivalent
  - If in some row  $P$  has a different truth value from  $Q$ , then  $P$  and  $Q$  are not logically equivalent

### De Morgan's Laws

- The negation of an and statement is logically equivalent to the or statement in which each component is negated.
- The negation of an or statement is logically equivalent to the and statement in which each component is negated

### Tautologies and Contradictions

**Definition** A **tautology** is a statement form that is always true regardless of the truth values of the individual statements substituted for its statement variables. A statement whose form is a tautology is a **tautological statement**. A **contradiction** is a statement form that is always false regardless of the truth values of the individual statements substituted for its statement variables. A **contradiction** is a statement form that is always false regardless of the truth values of the individual statements substituted for its statement variables. A statement whose form is a contradiction is a **contradictory statement**.

## Summary of Logical Equivalences

<b>Theorem 1.1.1 Logical Equivalences</b> Given any statement variables $p$ , $q$ , and $r$ , a tautology $t$ and a contradiction $c$ , the following logical equivalences hold		
<b>Commutative laws</b>	$p \wedge q \equiv q \wedge p$	$p \vee q \equiv q \vee p$
<b>Associative laws</b>	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	$(p \vee q) \vee r \equiv p \vee (q \vee r)$
<b>Distributive laws</b>	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
<b>Identity law</b>	$p \wedge t \equiv p$	$p \vee c \equiv p$
<b>Negation laws</b>	$p \vee \sim p \equiv t$	$p \wedge \sim p \equiv c$
<b>Double negative law</b>	$\sim(\sim p)$	
<b>Idempotent laws</b>	$p \wedge p \equiv p$	$p \vee p \equiv p$
<b>Universal bound laws</b>	$p \vee t \equiv t$	$p \wedge c \equiv c$
<b>DeMorgan's Laws</b>	$\sim(p \wedge q) \equiv \sim p \vee \sim q$	$\sim(p \vee q) \equiv \sim p \wedge \sim q$
<b>Absorption laws</b>	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$
<b>Negation of <math>t</math> and <math>c</math></b>	$\sim t \equiv c$	$\sim c \equiv t$

## 1.2 Conditional Statements

**Definition** If  $p$  and  $q$  are statement variables, the **conditional** of  $q$  and  $p$  is “If  $p$  then  $q$ ” or “ $p$  implies  $q$ ” and is denoted  $p \rightarrow q$ . It is false when  $p$  is true and  $q$  is false; otherwise it is true. We call  $p$  the **hypothesis** (or **antecedent**) of the conditional and  $q$  the **conclusion** (or **consequent**)

A statement that is true by virtue of the fact that its hypothesis is false is vacuously true or **true by default**.

### Order of operations

- $\sim$
- $\wedge$  and  $\vee$
- $\rightarrow$

### Logical Equivalences involving $\rightarrow$

Use truth tables to show logical equivalence of statements for  $\rightarrow$

### Negation of a Conditional Statement

The negation of “if  $p$  then  $q$ ” is logically equivalent to “ $p$  and not  $q$ ” which can be restated as  $\sim(p \rightarrow q) \equiv p \wedge \sim q$

### Contrapositive of a Conditional Statement

**Definition** The **contrapositive** of a conditional statement of the form “if  $p$  then  $q$ ” is  
If  $\sim q$  then  $\sim p$

Symbolically,

The contrapositive of  $p \rightarrow q$  is  $\sim q \rightarrow \sim p$

A conditional statement is logically equivalent to its contrapositive

### Converse and Inverse of a Conditional Statement

**Definition** Suppose a conditional statement of the form “If  $p$  then  $q$ ” is given.  
The **converse** is “If  $q$  then  $p$ ”  
The **inverse** is “If  $\sim p$  then  $\sim q$ ”

Symbolically,

The converse of  $p \rightarrow q$  is  $q \rightarrow p$   
The inverse of  $p \rightarrow q$  is  $\sim p \rightarrow \sim q$

### Only If and the Biconditional

**Definition** If  $p$  and  $q$  are statements,  
 $p$  **only if**  $q$  means “if not  $q$  then not  $p$ ”  
Or, equivalently,  
“if  $p$  then  $q$ ”

**Definition** Given statement variables  $p$  and  $q$ , the **biconditional** of  $p$  and  $q$  is “ $p$  if, and only if  $q$ ” and is denoted  $p \leftrightarrow q$ . It is true if both  $p$  and  $q$  have the same truth values and is false if  $p$  and  $q$  have opposite truth values. The words if and only if are sometimes abbreviated **iff**.

### Necessary and Sufficient Conditions

**Definition** If  $r$  and  $s$  are statements:  
 $r$  is a **sufficient condition** for  $s$  means “if  $r$  then  $s$ ”  
 $r$  is a **necessary condition** for  $s$  means “if not  $r$  then not  $s$ ”

$r$  is a necessary condition for  $s$  also means “if  $s$  then  $r$ ”

$r$  is a necessary and sufficient condition for  $s$  means “ $r$  if, and only if,  $s$ ”

### Remarks

In logic, a hypothesis and conclusion are not required to have related subject matters

In informal language, simple conditionals are often used to mean biconditionals



### 1.3 Valid and Invalid Arguments

**Definition** An **argument** is a sequence of statements, and an **argument form** is a sequence or statement forms. All statements in an argument and all statement forms in an argument form, except for the final one, are called **premises** (or **assumptions** or **hypotheses**). The final statement or statement form is called the **conclusion**. The symbol,  $\therefore$ , which is read “therefore,” is normally placed just before the conclusion. To say that an argument form is **valid** means that no matter what particular statements are substituted for the statement variables in its premises, if the resulting premises are all true, then the conclusion is also true. To say that an argument is **valid** means that its form is valid.

#### Testing an Argument Form for Validity

- Identify the premises and conclusion of the argument form
- Construct a truth table showing the truth values of all the premises and the conclusion
- If the truth table contains a row in which all the premises are true and the conclusion is false, then it is possible for an argument of the given form to have true premises and a false conclusion, and so the argument form is invalid. Otherwise, in every case where all the premises are true, the conclusion is also true, and so the argument form is valid.

#### Modus Ponens and Modus Tollens

An argument form consisting of two premises and a conclusion is called a syllogism. The first and second premises are called the major and minor premises, respectively.

##### Modus Ponens:

$$\begin{array}{l} \text{If } p \text{ then } q \\ p \\ \therefore q \end{array}$$

##### Modus Tollens:

$$\begin{array}{l} \text{If } p \text{ then } q \\ \sim p \\ \therefore \sim q \end{array}$$

#### Additional Valid Argument Forms: Rules of Inference

**Rule of inference** is a form of argument that is valid.

#### Fallacies

A **fallacy** is an error in reasoning that results in an invalid argument  
Using ambiguous premises and treating them as if they were unambiguous  
Begging the question - assuming what is to be proved without having derived it from the premises  
Jumping to a conclusion (without adequate grounds)  
Converse error – the converse of a statement is not logically equivalent to the statement

Inverse error – the inverse of a statement is not logically equivalent to the statement

### *Contradictions and Valid Arguments*

Contradiction rule – If you can show that the supposition that statement  $p$  is false leads logically to a contradiction, then you can conclude that  $p$  is true.

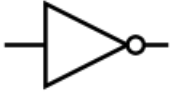



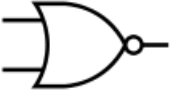
### *Summary of Rules of Inference*

<b>Valid Argument Forms</b>				
Modus Ponens	$p \rightarrow q$ $p$ $\therefore q$	Elimination	$p \vee q$ $\sim q$ $\therefore p$	$p \vee q$ $\sim p$ $\therefore q$
Modus Tollens	$p \rightarrow q$ $\sim q$ $\therefore \sim p$	Transitivity	$p \rightarrow q$ $q \rightarrow r$ $\therefore p \rightarrow r$	
Generalization	$p$ $\therefore p \vee q$	$q$ $\therefore p \vee q$	Proof by Division into Cases	$p \vee q$ $p \rightarrow r$ $q \rightarrow r$ $\therefore r$
Specialization	$p \wedge q$ $\therefore p$	$p \wedge q$ $\therefore q$		
Conjunction	$p$ $q$ $\therefore p \wedge q$	Contradiction Rule	$\sim p \rightarrow c$ $\therefore p$	

## 1.4 Application: Digital Logic Circuits

### Black Boxes and Gates:

Think of certain circuits as black boxes with an input and output

Type of Gate	Symbolic Representation	Action																		
Not		<table border="1"> <thead> <tr> <th>Input</th> <th>Output</th> </tr> </thead> <tbody> <tr> <td>P</td> <td>R</td> </tr> <tr> <td>1</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> </tr> </tbody> </table>	Input	Output	P	R	1	0	0	1										
Input	Output																			
P	R																			
1	0																			
0	1																			
And		<table border="1"> <thead> <tr> <th colspan="2">Input</th> <th>Output</th> </tr> <tr> <th>P</th> <th>Q</th> <th>R</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table>	Input		Output	P	Q	R	1	1	1	1	0	0	0	1	0	0	0	0
Input		Output																		
P	Q	R																		
1	1	1																		
1	0	0																		
0	1	0																		
0	0	0																		
Or		<table border="1"> <thead> <tr> <th colspan="2">Input</th> <th>Output</th> </tr> <tr> <th>P</th> <th>Q</th> <th>R</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table>	Input		Output	P	Q	R	1	1	1	1	0	1	0	1	1	0	0	0
Input		Output																		
P	Q	R																		
1	1	1																		
1	0	1																		
0	1	1																		
0	0	0																		
Nand		<table border="1"> <thead> <tr> <th colspan="2">Input</th> <th>Output</th> </tr> <tr> <th>P</th> <th>Q</th> <th>R=P Q</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> </tbody> </table>	Input		Output	P	Q	R=P Q	1	1	0	1	0	1	0	1	1	0	0	1
Input		Output																		
P	Q	R=P Q																		
1	1	0																		
1	0	1																		
0	1	1																		
0	0	1																		
Nor		<table border="1"> <thead> <tr> <th colspan="2">Input</th> <th>Output</th> </tr> <tr> <th>P</th> <th>Q</th> <th>R=P↓Q</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> </tbody> </table>	Input		Output	P	Q	R=P↓Q	1	1	1	1	0	1	0	1	1	0	0	0
Input		Output																		
P	Q	R=P↓Q																		
1	1	1																		
1	0	1																		
0	1	1																		
0	0	0																		

### Rules for a Combinational Circuit

- Never combine two input wires
- A single input wires can be split partway and used as input for two separate gates
- An output wire can be used as an input wire
- No output of a gate can eventually feed back into the gate

### Boolean Expression Corresponding to a Circuit

Boolean Variable – any variable, such as an input signal or a statement variable, that can only have two values

Boolean Expression – an expression composed of Boolean variables and their connectives

**Definition** A **recognizer** is a circuit that outputs a 1 for exactly one particular combination of input signals and outputs 0's for all other combinations

P	Q	R	$(P \wedge Q) \wedge \sim R$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

### Finding a Circuit That Corresponds to a Given Input/Output Table

Disjunctive normal form or sum-of-products – expression that is a disjunction of terms that are themselves conjunctions in which one of P or  $\sim P$ , one of Q or  $\sim Q$ , and one of R or  $\sim R$  all appear.

### Simplifying Combinational Circuits

**Definition** Two digital logic circuits are **equivalent** if and only if their input/output tables are identical

## 1.5 Application: Number Systems and Circuits for Addition

### Binary Representation of Numbers

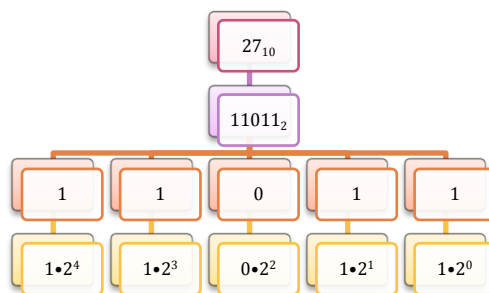
In decimal notation any positive integer can be written uniquely as a sum of products of the form

$$d \cdot 10^n$$

Any integer can be represented uniquely as a sum of products of the form

$$d \cdot 2^n$$

Where n is an integer and d is one of the binary digits or bits 0 or 1



## Binary addition and Subtraction

### Binary addition rules

- $0 + 0 = 0$
- $0 + 1 = 1$
- $1 + 0 = 1$
- $1 + 1 = 0$ , and carry 1 to the next more significant bit

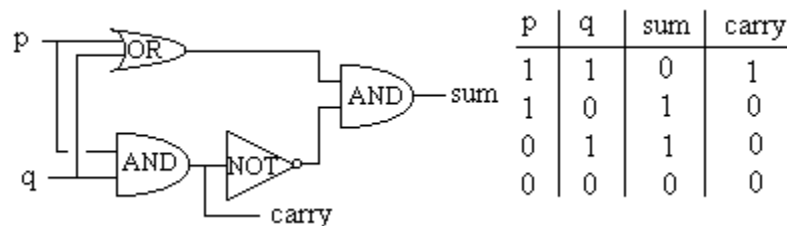
### Binary Subtraction Rules

- $0 - 0 = 0$
- $0 - 1 = 1$ , and borrow 1 from the next more significant bit
- $1 - 0 = 1$
- $1 - 1 = 0$

### Circuits for Computer Addition

$$\begin{aligned} 1_2 + 1_2 &= 10_2 \\ 1_2 + 0_2 &= 1_2 = 01_2 \\ 0_2 + 1_2 &= 1_2 = 0_2 \\ 0_2 + 0_2 &= 0_2 = 00_2 \end{aligned}$$

The circuit must have two outputs - one for the left binary digit (carry) and one for the right binary digit (sum). The sum can be produced using a circuit that corresponds to the Boolean equation  $(P \vee Q) \wedge \sim (P \wedge Q)$ . The circuit is called a half-adder.



Issue how to add three binary digits? Incorporate a circuit that will compute the sum of three binary digits called a full adder. Where P and Q are added and the result is added to R

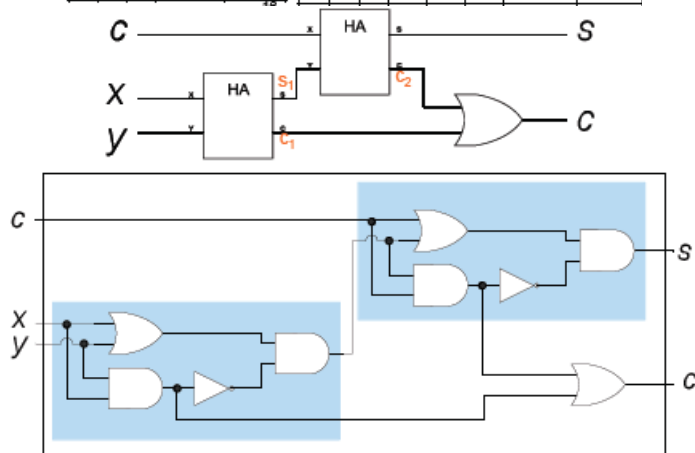
**Step 1:** Add P and Q using a half adder to obtain a binary number with two digits

$$\begin{array}{r} P \\ + \quad Q \\ \hline C_1 \quad S_1 \end{array}$$

**Step 2:** Add R to the sum  $C_1S_1$  by adding R to  $S_1$  using a half adder to obtain  $C_2S$  where S is the right most digit of the entire sum of P, Q, and R. C ( the left most bit) is 1 if and only if  $C_1=1$  or  $C_2=1$

$$\begin{array}{r} C_1 \quad S_1 \\ + \quad R \\ \hline C_2 \quad S \end{array}$$

x	y	c	carry	sum	x	y	c	s <sub>1</sub>	c <sub>1</sub>	c <sub>2</sub>	carry	sum
1	1	1	1	1	1	1	1	0	1	0	1	1
1	1	0	1	0	1	1	0	0	1	0	1	0
1	0	1	1	0	1	0	1	1	0	1	1	0
1	0	0	0	1	1	0	0	1	0	0	0	1
0	1	1	1	0	0	1	1	1	0	1	1	0
0	1	0	0	1	0	1	0	1	0	0	0	1
0	0	1	0	1	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0



### Hexadecimal Notation

Hexadecimal Notation, base 16 notation, is notation based on the fact any integer can be uniquely expressed as a sum of numbers in the form

$$d \cdot 16^n$$

Decimal	Hexadecimal	4-Bit Binary Equivalent
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

## Chapter 2 The Logic of Quantified Statements

---

### 2.1 Introduction to Predicates and Quantified Statements I

**Definition** a **predicate** is a sentence that contains a finite number of variables and becomes a statement when specific values are substituted for the variables. The **domain** of a predicate variable is the set of all values that may be substituted in place of the variable.

**Definition** if  $P(x)$  is a predicate and  $x$  has domain  $D$ , the **truth set** of  $P(x)$  is the set of all elements of  $D$  that make  $P(x)$  true when they are substituted for  $x$ . The truth set of  $P(x)$  is denoted

$$\{x \in D | P(x)\}$$

Which is read, “the set of all  $x$  in  $D$  such that  $P(x)$ ”.

#### **The Universal Quantifier: $\forall$**

The symbol  $\forall$  denotes “for all” and is called **the universal quantifier**.

**Definition** Let  $Q(x)$  be a predicate and  $D$  the domain of  $x$ . A **universal statement** is a statement of the form “ $\forall x \in D, Q(x)$ .” It is defined to be true if and only if  $Q(x)$  is true for every  $x$  in  $D$ . It is defined to be false, if, and only if,  $Q(x)$  is false for at least one  $x$  in  $D$ . A value for  $x$  for which  $Q(x)$  is false is called a **counterexample** to the universal statement.

The technique used to show the truth of the universal statement is a method called the **method of exhaustion** – showing for each individual element of the domain the predicate is true.

#### **The Existential Quantifier: $\exists$**

The symbol  $\exists$  denotes “there exists” and is called the existential quantifier.

**Definition** Let  $Q(x)$  be a predicate and  $D$  the domain of  $x$ . An **existential statement** is a statement of the form “ $\exists x \in D, Q(x)$ .” It is defined to be true if and only if  $Q(x)$  is true for at least one  $x$  in  $D$  such that  $Q(x)$ . It is defined to be false, if, and only if,  $Q(x)$  is false for all  $x$  in  $D$ .

#### **Universal Conditional Statements**

Universal conditional statement is a statement in the form of  $\forall x$ , if  $P(x)$  then  $Q(x)$

#### **Equivalent Forms of Universal and Existential Statements**

$$\forall x \in U, \text{ if } P(x) \text{ then } Q(x) \equiv \forall x \in D, Q(x)$$

$$\forall x \in D, Q(x) \equiv \forall x, \text{ if } x \text{ is in } D \text{ then } Q(x)$$



### Implicit Quantification

**Notation** Let  $P(x)$  and  $Q(x)$  be predicates and suppose the common domain of  $x$  is  $D$ . The notation  $P(x) \Rightarrow Q(x)$  means that every element in the truth set of  $P(x)$  is in the truth set of  $Q(x)$ , or equivalently,  $\forall x, P(x) \rightarrow Q(x)$ . The notation  $P(x) \Leftrightarrow Q(x)$  means that  $P(x)$  and  $Q(x)$  have identical truth sets, or equivalently  $\forall x, P(x) \leftrightarrow Q(x)$ .

## 2.2 Introduction to Predicates and Quantified Statements II

### Negation of Quantified Statements

#### Theorem 2.2.1 Negation of a Universal Statement

The negation of a statement of the form

$$\forall x \text{ in } D, Q(x)$$

Is logically equivalent to a statement of the form

$$\exists x \text{ in } D \text{ such that } \sim Q(x).$$

Symbolically,

$$(\sim \forall x \in D, Q(x)) \equiv \exists x \in D \text{ such that } \sim Q(x)$$

#### Theorem 2.2.2 Negation of an Existential Statement

The negation of a statement of the form

$$\exists x \text{ in } D \text{ such that } Q(x)$$

Is logically equivalent to a statement of the form

$$\forall x \text{ in } D, \sim Q(x)$$

Symbolically,

$$\sim(\exists x \in D \text{ such that } Q(x)) \equiv \forall x \in D, \sim Q(x)$$

### Negations of Universal Conditional Statements

$$\sim(\forall x, P(x) \rightarrow Q(x)) \equiv \exists x \text{ such that } \sim(P(x) \rightarrow Q(x))$$

$$\sim(P(x) \rightarrow Q(x)) \equiv P(x) \wedge \sim Q(x)$$

$$\sim(\forall x, P(x) \rightarrow Q(x)) \equiv \exists x \text{ such that } P(x) \wedge \sim Q(x)$$

$$\sim(\forall x, P(x) \text{ then } Q(x)) \equiv \exists x \text{ such that } P(x) \text{ and } \sim Q(x)$$

### The Relation among $\forall, \exists, \wedge$ , and $\vee$

If  $Q(x)$  is a predicate and the domain  $D$  of  $x$  is the set  $\{x_1, x_2, \dots, x_n\}$  then the statements

$$\forall x \in D, Q(x)$$

And

$$Q(x_1) \wedge Q(x_2) \wedge \dots \wedge Q(x_n)$$

Are logically equivalent

If  $Q(x)$  is a predicate and the domain  $D$  of  $x$  is the set  $\{x_1, x_2, \dots, x_n\}$  then the statements

$$\exists x \in D, Q(x)$$

And

$$Q(x_1) \vee Q(x_2) \vee \dots \vee Q(x_n)$$

Are logically equivalent

### Variants of Universal Conditional Statements

#### Definition

Consider a statement of the form

$$\forall x \in D, \text{ if } P(x) \text{ then } Q(x)$$

Its **contrapositive** is the statement

$$\forall x \in D, \text{ if } \sim Q(x) \text{ then } \sim P(x)$$

Its **converse** is the statement

$$\forall x \in D, \text{ if } Q(x) \text{ then } P(x)$$

Its **inverse** is the statement

$$\forall x \in D, \text{ if } \sim P(x) \text{ then } \sim Q(x)$$

### Necessary and Sufficient Conditions, Only If

#### Definition

$\forall x, r(x)$  is a **sufficient condition** for  $s(x)$  means  $\forall x$  if  $r(x)$  then  $s(x)$

$\forall x, r(x)$  is a **necessary condition** for  $s(x)$  means  $\forall x$ , if  $\sim r(x)$  then  $\sim s(x)$  or equivalently  $\forall x$ , if  $s(x)$  then  $r(x)$

$\forall x, r(x)$  **only if**  $s(x)$  means  $\forall x$ , if  $\sim s(x)$  then  $\sim r(x)$  or equivalently  $\forall x$ , if  $r(x)$  then  $s(x)$

## 2.3 Statements Containing Multiple Quantifiers

### Negations of Multiply-Quantified Statements

$$\sim(\forall x \text{ in } D, \exists y \text{ in } E \text{ such that } P(x, y)) \equiv \exists x \text{ in } D \text{ such that } \forall y \text{ in } E, \sim P(x, y)$$

$$\sim(\exists x \text{ in } D, \forall y \text{ in } E \text{ such that } P(x, y)) \equiv \forall x \text{ in } D \text{ such that } \exists y \text{ in } E, \sim P(x, y)$$

### Order of Quantifiers

If a statement contains two different quantifiers reversing the order of the quantifiers can change the truth value of the statement to its opposite

If one quantifier immediately follows another quantifier of the same type, then the order of quantifier does not affect the meaning.

### Prolog

A simple prolog program consists of a set of statements describing some situation together with questions about the situation. Built into the language are search and inference techniques needed to answer the question

$$\text{isabove}(x, y) = x \text{ is above } y$$

$$\text{color}(x, \text{color}) = x \text{ is } \text{color}$$

$$? \text{isabove}(x, y) = \text{is } x \text{ above } y?$$

$$? \text{color}(x, \text{color}) = \text{is } x \text{ } \text{color}?$$

## Chapter 3 Elementary Number Theory and Methods of Proof

### 3.1 Direct Proof and Counterexample I: Introduction

In order to evaluate the truth or falsity of a statement, you must understand what the statement is about. You know the meanings of all the terms that occur in the statement

**Definitions** An integer  $n$  is **even** if, and only if,  $n$  equals twice some integer. An integer  $n$  is **odd** if, and only if,  $n$  equals twice some integer plus 1. Symbolically, if  $n$  is an integer, then

$$\begin{aligned}n \text{ is even} &\Leftrightarrow \exists \text{ an integer } k \text{ such that } n = 2k \\n \text{ is odd} &\Leftrightarrow \exists \text{ an integer } k \text{ such that } n = 2k + 1\end{aligned}$$

**Definition** An integer  $n$  is **prime** if, and only if,  $n > 1$  and for all positive integers  $r$  and  $s$ , if  $n = r \cdot s$ , then  $r = 1$  or  $s = 1$ . An integer  $n$  is **composite** if, and only if,  $n > 1$  and  $n = r \cdot s$  for some positive integers  $r$  and  $s$  with  $r \neq 1$  and  $s \neq 1$ . It follows that if  $n$  is an integer greater than 1, then

$$\begin{aligned}n \text{ is prime} &\Leftrightarrow \forall \text{ positive integers } r \text{ and } s, \text{ if } n = r \cdot s \text{ then } r = 1 \text{ or } s = 1 \\n \text{ is composite} &\Leftrightarrow \forall \text{ positive integers } r \text{ and } s, \text{ if } n = r \cdot s \text{ then } r \neq 1 \text{ and } s \neq 1\end{aligned}$$

#### Proving Existential Statements

To prove a statement in the form

$$\exists x \in D, Q(x)$$

**Constructive Proofs of Existence:** Find an  $x$  for which  $Q(x)$  is true or provide a set of directions for finding an  $x$  such that  $Q(x)$  is true.

**Non-constructive proof of Existence:** showing either (a) that the existence of a value  $x$  that makes  $Q(x)$  true is guaranteed by an axiom or a previously proved theorem or (b) that the assumption that there is no such  $x$  leads to a contradiction

#### Disproving Universal Statements by Counterexample

##### Disproof by Counterexample

To disprove a statement of the form “ $\forall x \in D$ , if  $P(x)$  then  $Q(x)$ ,” find a value  $x$  in  $D$  for which  $P(x)$  is true and  $Q(x)$  is false. Such an  $x$  is called a **counterexample**

#### Proving Universal Statements

##### Method of Generalizing from the Generic Particular

To show that every element of a domain satisfies a certain property, suppose  $x$  is a particular but arbitrary chosen element of the domain, and show that  $x$  satisfies the property.

### Method of Direct Proof

1. Express the statement to be proved in the form " $\forall x \in D, P(x) \rightarrow Q(x)$ "
2. Start the proof by supposing  $x$  is a particular by arbitrarily chosen element of  $D$  for which the hypothesis  $P(x)$  is true
3. Show that the conclusion  $Q(x)$  is true by using definitions, previously established results and the rules for logical inference.

### Directions for Writing Proofs of Universal Statements

1. Copy the statement of the theorem to be proved on your paper
2. Clearly mark the beginning of your proof with the word proof
3. Make your proof self contained (identify each variable used in the proof)
4. Write your proof in complete sentences
5. Give a reason for each assertion you make in your proof
6. Include the little words that make the logic of your arguments clear

### Common Mistakes

1. Arguing from examples
2. Using the same letter to mean two different things
3. Jumping to a conclusion
4. Begging the question (assuming what is to be proved)
5. Misuse of the word if

### Showing that an Existential Statement is False

Because the negation of an existential statement is a universal statement, to prove an existential statement is false you must prove that its negation to be true.

## 3.2 Direct Proof and Counterexample II: Rational Numbers

**Definition** a **real number** is rational if, and only if, it can be expressed as a quotient of two integers with a nonzero denominator. A real number that is not rational is **irrational**. More formally, if  $r$  is a real number, then

$$r \text{ is rational} \Leftrightarrow \exists \text{ integers } a \text{ and } b \text{ such that } r = \frac{a}{b} \text{ and } b \neq 0$$

## 3.3 Direct Proof and Counter Example III: Divisibility

**Definition** If  $n$  and  $d$  are integers, then

$n$  is **divisible by**  $d$  if, and only if,  $n = dk$  for some integer  $k$

Alternatively, we say that

$n$  is a **multiple of**  $d$

$d$  is a **factor of**  $n$

$d$  is a **divisor of**  $n$

$d$  **divides**  $n$

The notation  $d|n$  is read " $d$  divides  $n$ ." Symbolically, if  $n$  and  $d$  are integers

$$d|n \Leftrightarrow \exists \text{ an integer } k \text{ such that } n = dk$$

For all integers  $n$  and  $d$  with  $d \neq 0$ ,  $d \nmid n \Leftrightarrow \frac{n}{d}$  is not an integer

### The Unique Factorization Theorem

#### Theorem 3.3.3 Unique Factorization Theorem for the Integers (Fundamental Theorem of Arithmetic)

Given any integer  $n > 1$ , there exist a positive integer  $k$ , distinct prime numbers  $(p_1, p_2, \dots, p_k)$  and positive integers  $(e_1, e_2, \dots, e_k)$  such that

$$n = p_1^{e_1} p_2^{e_2} p_3^{e_3} \cdots p_k^{e_k}$$

And any other expression of  $n$  as a product of prime numbers is identical to this except, perhaps, for the order in which the factors are written

**Definition** Given any integer  $n > 1$ , the **standard factored form** of  $n$  is an expression of the form

$$n = p_1^{e_1} p_2^{e_2} p_3^{e_3} \cdots p_k^{e_k}$$

Where  $k$  is a positive integer;  $(p_1, p_2, \dots, p_k)$  are prime numbers;  $(e_1, e_2, \dots, e_k)$  are positive integers; and  $p_1 < p_2 < \cdots < p_k$

### 3.4 Direct Proof and Counterexample IV: Division into Cases and the Quotient-Remainder Theorem

### 3.5 Direct Proof and Counterexample V: Floor and Ceiling

### 3.6: Indirect Argument: Contradiction and Contraposition

#### Method by Proof by Contradiction

1. Suppose the statement to be proved is false. That is, suppose the negation of the statement is true
2. Show that this supposition leads logically to a contradiction
3. Conclude that the statement to be proved is true

#### Theorem 3.6.1

There is no greatest integer.

#### Proof:

[We take the negation of the theorem and suppose it to be true.] Suppose not. That is suppose there is a greatest integer  $N$ . [We must deduce a contradiction.] Then  $N \geq n$  for every integer  $n$ . Let  $M = N + 1$ . Now  $M$  is an integer since it is a sum of integers. Also  $M > N$  since  $M = N + 1$ .

Thus  $M$  is an integer that is greater than  $N$ . So  $N$  is the greatest integer and  $N$  is not the greatest integer, which is a contradiction. [This contradiction shows that supposition is false and, hence that the theorem is true.]

### Theorem 3.6.2

There is no integer that is both even and odd.

#### Proof:

[We take the negation of the theorem and suppose it to be true.] Suppose not. That is, suppose there is an integer  $n$  that is both even and odd. [we must deduce a contradiction.] By definition of even,  $n = 2a$  for some integer  $a$ , and by definition of odd,  $n = 2b + 1$  for some integer  $b$ . Consequently, by equating the two expressions for  $n$

$$2a = 2b + 1$$

And so

$$2a - 2b = 1$$

$$2(a - b) = 1$$

$$(a - b) = \frac{1}{2}$$

Now since  $a$  and  $b$  are integers, the difference  $a - b$  must also be an integer. But  $a - b = \frac{1}{2}$ , and  $\frac{1}{2}$  is not an integer. Thus  $a - b$  is an integer and  $a - b$  is not integer, which is a contradiction. [This contradiction shows that the supposition is false and, hence, that the theorem is true.]

### Theorem 3.6.3

The sum of any rational number and any irrational number is irrational

#### Proof:

[We take the negation of the theorem and suppose it to be true.] Suppose not. That is, suppose there is a rational number  $r$  and an irrational number  $s$  such that  $r+s$  is rational. [we must deduce a contradiction.] By definition of rational,  $r=a/b$  and  $r+s=c/d$  for some integers  $a,b,c$ , and  $d$  with  $b \neq 0$  and  $d \neq 0$ . By substitution

$$\frac{a}{b} + s = \frac{c}{d}$$

And so

$$s = \frac{c}{d} - \frac{a}{b} = \frac{bc - ad}{bd}$$

Now  $bc - ad$  and both  $bd$  are integers [since  $a,b,c$ , and  $d$  are, and since products and differences of integers are integers] and  $bd \neq 0$  [by the zero product property]. Hence  $s$  is a quotient of the two integers  $bc - ad$  and  $bd \neq 0$ . Thus, by definition of rational  $s$  is rational, which contradicts the supposition that  $s$  is irrational. [Hence, the supposition is false and the theorem is true.]

## Argument by Contraposition

### Method of Proof by Contradiction

1. Express the statement to be proved in the form  
$$\forall x \in D, P(x) \rightarrow Q(x)$$
2. Rewrite this statement in the contrapositive form  
$$\forall x \in D, \sim Q(x) \rightarrow \sim P(x)$$
3. Prove the contrapositive by a direct proof  
Suppose  $x$  is a particular by arbitrarily chosen element of  $D$  such that  $Q(x)$  is false  
Show that  $P(x)$  is false

**Proposition 3.6.4:** For all integers  $n$ , if  $n^2$  is even then  $n$  is even

#### Proof (by contraposition):

Suppose  $n$  is any odd integer. [We must show that  $n^2$  is odd.] By definition of odd,  $n = 2k + 1$  for some integer  $k$ . By substitution and algebra,  $n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$ . But  $2k^2 + 2k$  is an integer because products and sums of integers are integers. So  $n^2 = (\text{an integer}) + 1$ , and thus, by definition of odd,  $n^2$  is odd [as was to be shown.]

#### Proof (by contradiction):

[We take the negation of the theorem and suppose it to be true.] Suppose not. That is, suppose there is an integer  $n$  such that  $n^2$  is even and  $n$  is not even. [We must deduce a contradiction.] By the quotient-remainder theorem with  $d=2$ , any integer is even or odd. Hence, since  $n$  is not even it is odd, and thus by definition of odd,  $n=2k+1$  for some integer  $k$ . by substitution and algebra:  $n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$ . But  $2k^2 + 2k$  is an integer because products and sums of integers are integers. So  $n^2 = (\text{an integer}) + 1$ , and thus, by definition of odd,  $n^2$  is odd. Therefore,  $n^2$  is both even and odd. This contradicts Theorem 3.6.2, which states that no integer can be both even and odd. [This contradiction shows that the supposition is false and, hence, that the proposition is true.]

## Relation between Proof by Contradiction and Proof by Contraposition

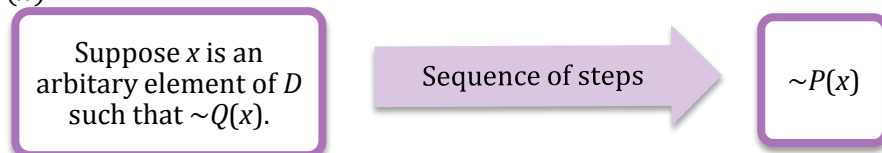
In a proof by contraposition, the statement

$$\forall x \in D, P(x) \rightarrow Q(x)$$

Is proved by giving a direct proof of the equivalent statement

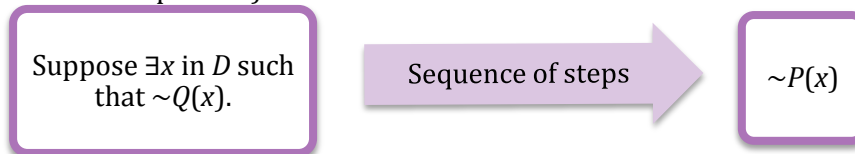
$$\forall x \in D, \sim Q(x) \rightarrow \sim P(x)$$

To do this you are given an arbitrary element  $x$  of  $D$  such that  $\sim Q(x)$ . Then show that  $\sim P(x)$ .



Exactly the same sequence of steps can be used as the heart of a proof by contradiction for the given statement. The only thing that changes is the context in which the steps are written down.

To rewrite the proof by contradiction, you suppose there is an  $x$  in  $D$  such that  $P(x)$  and  $\sim Q(x)$ . You then follow the steps of the proof by contraposition do deduce the statement  $\sim P(x)$ . But  $\sim P(x)$  is a contradiction to the supposition that  $P(x)$  and  $\sim Q(x)$ . (Because to contradict a conjunction of statement, it is only necessary to contradict one component.)



### 3.7 Two Classical Theorems

#### The Irrationality of $\sqrt{2}$

**Theorem 3.7.1 Irrationality of  $\sqrt{2}$ :**  $\sqrt{2}$  is irrational

**Proof:**

[we take the negation and suppose it to be true.] Suppose not. That is suppose  $\sqrt{2}$  is rational. Then there are integers  $m$  and  $n$  with no common factors such that

$$\sqrt{2} = \frac{m}{n}$$

[By dividing  $m$  and  $n$  by any common factors if necessary]. [We must derive a contradiction.] Squaring both sides of the equation gives us

$$2 = \frac{m^2}{n^2}$$

Or, equivalently

$$m^2 = 2n^2$$

Note that equation implies that  $m^2$  is even (by the definition of even). It follows that  $m$  is even (by proposition 3.6.4). We file this fact away for future reference and also deduce (by definition of even) that

$$m = 2k$$

Substituting  $m = 2k$ , we see that

$$m^2 = (2k)^2 = 4k^2 = 2n^2$$

Dividing both sides of the right most equation by 2 gives

$$n^2 = 2k^2$$

Consequently,  $n^2$  is even, and so  $n$  is even (by proposition 3.6.4). But we also know that  $m$  is even. Hence both  $m$  and  $n$  have a common factor of 2. But this contradicts the supposition that  $m$  and  $n$  have no common factors. [Hence the supposition is false and so the theorem is true].



### Proposition 3.7.3

For any Integer  $a$  and any prime number  $p$ , if  $p|a$  then  $p \nmid (a + 1)$ .

#### Proof:

Suppose not. That is suppose there exists an integer  $a$  and a prime number  $p$  such that  $p|a$  and  $p|(a + 1)$ . Then by definition of divisibility, there exists integers  $r$  and  $s$  such that  $a = pr$  and  $a + 1 = ps$ . It follows that  $1 = (a + 1) - a = ps - pr = p(s - r)$ , and so (since  $s - r$  is an integer)  $p|1$ . But the only integer divisors of 1 are 1 and -1, and since  $p$  is prime,  $p > 1$ . Thus  $p \leq 1$  and  $p > 1$ , which is a contradiction. [Hence the supposition is false and the proposition is true.]

### Theorem 3.7.4 Infinitude of the Primes

The set of prime numbers is infinite

#### Proof (by contradiction):

Suppose not. Suppose the set of prime numbers is finite. [We must deduce a contradiction.] The all the prime numbers can be listed, say, in ascending order:

$$p_1 = 2, p_2 = 3, p_3 = 5, \dots, p_n$$

Consider the integer

$$N = p_1 p_2 p_3 \cdots p_n + 1$$

Then  $N > 1$ , and so, by theorem 3.3.2  $N$  is divisible by some prime number  $p$ . Also, since  $p$  is prime,  $p$  must equal one of the prime numbers  $p_1, p_2, p_3, \dots, p_n$ . Thus  $p|(p_1 p_2 p_3 \cdots p_n)$ . By proposition 3.7.3  $p \nmid (p_1 p_2 p_3 \cdots p_n + 1)$ , and so  $p \nmid N$ . Hence  $p|N$  and  $p \nmid N$ , which is a contradiction. [Hence the supposition is false and the theorem is true.]

## 3.8 Application: Algorithms

### An Algorithmic Language

Variable – a specific storage location in a computer's memory

Data type – set in which the variable in which the variable takes its values

Assignments statement – gives a value to a variable

Conditional statements – allow using the current values of program variables to determine which algorithm statement will be executed next

**if** (*condition*)

**Then**  $s_1$

**else**  $s_2$

Or

**if**(*condition*)**then**  $s_1$

Where *condition* is a predicate involving algorithm variables and where  $s_1$  and  $s_2$  are algorithm statements or groups of algorithm statements

Execution of an **if-then-else** statement

1. The condition is evaluated by substituting the current values of all algorithms variables appearing in it and evaluating the true or falsity of the resulting statement
2. If the condition is true, then  $s_1$  is executed and execution moves to the next algorithm statement following the **if-then-else** statement
3. If condition is false, then  $s_2$  is executed and moves to the next algorithm statement following the **if-then-else** statement

Execution of an **if-then** statement is similar to execution of an **if-then-else** statement, except that if condition is false, execution passes immediately to the next algorithm statement following the **if-then** statement. Often condition is called a guard because it is stationed before  $s_1$  and  $s_2$  and restricts access of them

**Iterative Statements** are used when a sequence of algorithm statements is to be executed over and over again

### *A Notation for Algorithms*

We will generally include the following information when describing algorithms formally

1. The name of the algorithm, together with a list of input and output variables
2. A brief description of how the algorithm works
3. The input variable names, labeled by data type (whether integer, real number, and so forth)
4. The statements that make up the body of the algorithm, possibly with explanatory comments
5. The output variable names, labeled by data types

## The Division Algorithm

### Algorithm 3.8.1 Division Algorithm

[Given a nonnegative integer  $a$  and a positive integer  $d$ , the aim of the algorithm is to find integers  $q$  and  $r$  that satisfy the conditions  $a=dq+r$  and  $0 \leq r < d$ . This is done by subtracting  $d$  repeatedly from  $a$  until the result is less than  $d$  is still nonnegative

$$0 \leq a - d - d - d - \dots - d = a - dq < d$$

The total number of  $d$ 's that are subtracted in the quotient  $q$ . The quantity  $a-dq$  equals the remainder  $r$ .]

**Input:**  $a$  [a nonnegative integer],  $d$  [a positive integer]

#### Algorithm Body:

$r:=a, q:=0$

[Repeatedly subtract  $d$  from  $r$  until a number less than  $d$  is obtained. Add 1 to  $q$  each time  $d$  is subtracted]

**While** ( $r \geq d$ )

$r:=r-d$

$q:=q+1$

**end while**

[After execution of the **while** loop,  $a=dq+r$ ]

**Output:**  $q, r$  [nonnegative integers]

## The Euclidean Algorithm

**Definition** Let  $a$  and  $b$  be integers that are not both zero. The **greatest common divisor** of  $a$  and  $b$  denoted  $\gcd(a, b)$  is that integer  $d$  with the following properties:

1.  $d$  is a common divisor of both  $a$  and  $b$ . In other words  
 $d|a$  and  $d|b$
2. For all integers  $c$ , if  $c$  is a common divisor of both  $a$  and  $b$ , then  $c$  is less than or equal to  $d$ . In other words

$$\text{For all integers } c, \text{ if } c|a \text{ and } c|b, \text{ then } c \leq d$$

### Lemma 3.8.1

If  $r$  is a positive integer, then  $\gcd(r, 0)=r$

#### Proof:

Suppose  $r$  is a positive integer. [We must show that the greatest common divisor of both  $r$  and  $0$  is  $r$ .] Certainly,  $r$  is common divisor of both  $r$  and  $0$  because  $r$  divides itself and also  $r$  divides  $0$  (since every positive integer divides  $0$ ). Also no integer larger than  $r$  can be a common divisor of  $r$  and  $0$  (since no integer larger than  $r$  can divide  $r$ ). Hence  $r$  is the greatest common divisor of  $r$  and  $0$ .

**Lemma 3.8.2**

If  $a$  and  $b$  are any integers with  $b \neq 0$  and  $q$  and  $r$  are any integers such that

$$a = bq + r,$$

Then

$$\gcd(a, b) = \gcd(b, r)$$

**Algorithm 3.8.2 Euclidean Algorithm**

[Given two integers  $A$  and  $B$  with  $A > B \geq 0$ , this algorithm computes  $\gcd(A, B)$ . It is based on two facts:

1.  $\gcd(a, b) = \gcd(b, r)$  if  $a, b, q$ , and  $r$  are integers with  $a = b \cdot q + r$  and  $0 \leq r < b$
2.  $\gcd(a, 0) = a$ .]

**Input:**  $A, B$  [integers with  $A > B \geq 0$ ]

**Algorithm Body:**

$a := A, b := B, r := B$

[If  $b \neq 0$ , compute  $a \bmod b$ , the remainder of the integer division of  $a$  by  $b$ , and set  $r$  equal to this value. Then repeat the process using  $b$  in place of  $a$  and  $r$  in place of  $b$ .]

**While ( $b \neq 0$ )**

$r := a \bmod b$

[the value of  $a \bmod b$  can be obtained by calling the division algorithm.]

$a := b$

$b := r$

**end while**

[after execution of the while loop,  $\gcd(A, B) = a$ .]

$\gcd := a$

**Output:**  $\gcd$  [a positive integer]

## Chapter 4 Sequences and Mathematical Induction

### 4.1 Sequences

In this section, we define the term **sequence** informally as a set of elements written in a row. In the sequence denoted

$$a_m, a_{m+1}, a_{m+2}, \dots, a_n$$

Each individual element  $a_k$  is called a **term**. The  $k$  in  $a_k$  is called a subscript or index,  $m$  (which may be any integer) is the subscript of the **initial term**, and  $n$  is the subscript of the **final term**. The notation

$$a_m, a_{m+1}, a_{m+2}, \dots$$

denotes an **infinite sequence**. An **explicit formula** or **general formula** for a sequence is a rule that shows how the values of  $a_k$  depend on  $k$ .

#### Summation Notation

The notation

$$\sum_{k=1}^n a_k$$

is used to represent the sum in expanded form by

$$a_1 + a_2 + a_3 + \dots + a_n$$

More generally, if  $m$  and  $n$  are integers and  $m \leq n$ , then **the summation from  $k$  equals  $m$  to  $n$  of  $a_k$**  is the sum of all the terms  $a_m, a_{m+1}, a_{m+2}, \dots, a_n$ . We write

$$\sum_{k=m}^n a_k = a_m + a_{m+1} + a_{m+2} + \dots + a_n$$

And call  $k$  the **index** of the summation,  $m$  the **lower limit** of the summation, and  $n$  the **upper limit** of the summation.

#### Product Notation

The product from  $k$  equals  $m$  to  $n$  of  $a_k$  is the product of all the terms  $a_m, a_{m+1}, a_{m+2}, \dots, a_n$ . That is,

$$\prod_{k=m}^n a_k = a_m \cdot a_{m+1} \cdot a_{m+2} \cdots a_n$$

#### Factorial Notation

**Definition** for each positive integer  $n$ , the quantity  **$n$  factorial** denoted  $n!$ , is defined to be the product of all the integers from 1 to  $n$ :

$$n! = n \cdot (n-1) \cdots 3 \cdot 2 \cdot 1$$

Zero factorial, denoted  $0!$ , is defined to be one:

$$0! = 1.$$

### Properties of Summations and Products

**Theorem 4.1.1** If  $a_m, a_{m+1}, a_{m+2}, \dots$  and  $b_m, b_{m+1}, b_{m+2}, \dots$  are sequences of real numbers and  $c$  is any real number, then the following equations hold for any integer  $n \geq m$ :

$$\begin{aligned}\sum_{k=m}^n a_k + \sum_{k=m}^n b_k &= \sum_{k=m}^n (a_k + b_k) \\ c \cdot \sum_{k=m}^n a_k &= \sum_{k=m}^n c \cdot a_k \\ \left( \prod_{k=m}^n a_k \right) \cdot \left( \prod_{k=m}^n b_k \right) &= \prod_{k=m}^n (a_k \cdot b_k)\end{aligned}$$

### Sequences In Computer Programming

One-dimensional arrays – finite sequences

Example: find the sum of  $a[1], a[2], \dots, a[n]$

$s := a[1]$

**For**  $k := 2$  **to**  $n$

$s := s + a[k]$

**next**  $k$

## 4.2 Mathematical Induction I

### Principle of Mathematical Induction

Let  $P(n)$  be a property that is defined for integers  $n$ , and let  $a$  be a fixed integer.

Suppose the following two statements are true:

1.  $P(a)$  is true
2. For all integers  $k \geq a$ , if  $P(k)$  is true then  $P(k + 1)$  is true.

Then the statement

$$\text{for all integers } n \geq a, P(n)$$

is true.

### Method of Proof by Mathematical Induction

Consider a statement of the form, "For all integers  $n \geq a$ , a property  $P(n)$  is true."

To prove such a statement, perform the following two steps:

**Step 1 (Basis Step):** Show that the property is true for  $n = a$

**Step 2 (Inductive Step):** Show that for all integers  $k \geq a$ , if the property is true for  $n = k$  then it is true for  $n = k + 1$ . To perform this step,

**Suppose** that the property is true for  $n = k$ , where  $k$  is any particular but arbitrary chosen integers with  $k \geq a$ . [This supposition is called the **inductive hypothesis**]. Then **show** that the property is true for  $n = k + 1$

### Proposition 4.2.1

Let  $P(n)$  be the property " $n$  cents can be obtained using 3 and 5 cent coins." Then  $P(n)$  is true for all integers  $n \geq 8$ .

**Theorem 4.2.2 Sum of the First n Integers**

For all integers  $n \geq 1$

$$1 + 2 + \dots + n = \frac{n(n+1)}{2}$$

**Theorem 4.2.3 Sum of a Geometric Sequence**

For any real number  $r$  except 1, and any integer  $n \geq 0$ ,

$$\sum_{i=0}^n r^i = \frac{r^{n+1} - 1}{r - 1}$$

**4.3 Mathematical Induction II****Proposition 4.3.1**

For all integers  $n \geq 1$ ,  $2^{2n} - 1$  is divisible by 3

**Proposition 4.3.2**

For all integers  $n \geq 3$ ,  $2n + 1 < 2^n$ .

**4.4 Strong Mathematical Induction and the Well-Ordering Principle****Principle of Strong Mathematical Induction**

Let  $P(n)$  be a property that is defined for integers  $n$ , and let  $a$  and  $b$  be fixed integers with  $a \leq b$ . Suppose the following two statements are true.

1.  $P(a), P(a + 1), \dots$ , and  $P(b)$  are all true (basis step)
2. For any integer  $k > b$ , if  $P(i)$  is true for all integers  $i$  with  $a \leq i < k$ , then  $P(k)$  is true. (inductive step)

Then the statement

$$\text{for all integers } n \geq a, P(n)$$

is true. (The supposition that  $P(i)$  is true for all integers  $i$  with  $a \leq i < k$  is called the inductive hypothesis.)

**Binary Representation of Integers****Theorem 4.4.1 Existence and Uniqueness of Binary Integer Representations**

Given any positive integer  $n$ ,  $n$  has a unique representation in the form

$$n = c_r 2^r + c_{r-1} 2^{r-1} + \dots + c_2 2^2 + c_1 2^1$$

Where  $r$  is a nonnegative integer,  $c_r = 1$ , and  $c_j = 1$  or  $0$  for all  $j = 0, 1, 2, \dots, r - 1$ .

**The Well-Ordering Principle for the Integers****Well-Ordering Principle for the Integers**

Let  $S$  be a set containing one or more integers all of which are greater than some fixed integer. Then  $S$  has a least element.

**Quotient-Remainder Theorem (existence part)** Given any integer  $n$  and any positive integer  $d$ , there exist integers  $q$  and  $r$  such that

$$n = dq + r \text{ and } 0 \leq r < d$$

## 4.5 Application: Correctness of Algorithms

### Assertions

Consider an algorithm that is designed to produce a certain final state from a certain initial state. Both the initial and final states can be expressed as predicates involving the input and output variable. The predicate describing the initial state is called the **pre-condition** for the algorithm and the predicate describing the final state is called the **post-condition** for the algorithm.

### Loop Invariants

The method of loop invariants is used to prove correctness of a loop with respect to certain pre- and post- conditions. Suppose that an algorithm contains a while loop and that entry to this loop is restricted by a condition  $G$ , called the guard. Suppose also that assertions describing the current states of algorithm variables have been placed immediately preceding and immediately following the loop. The assertion just preceding the loop is called the pre-condition for the loop and the one just following is called the post-condition for the loop. The annotated loop has the following appearance:

```
[Pre-condition for the loop]
while ( $G$ )
    [Statements in the body of the loop. None contain branching
    statements that lead outside the loop.]
end while
[Post-condition for the loop]
```

**Definition** A loop is defined as correct with respect to its **pre- and post-conditions** if, and only if, whenever the algorithm variables satisfy the precondition for the loop and the loop terminates after a finite number of steps, the algorithm variables satisfy the post-condition for the loop.

A **loop invariant** is a predicate with domain a set of integers, which satisfies the condition: for each iteration of the loop, if the predicate is true before the iteration, then it is true after the iteration. Furthermore, if the predicate satisfies the following two additional conditions, the loop will be correct with respect to its pre- and post-conditions:

1. It is true before the first iteration of the loop
2. If the loop terminates after a finite number of iterations, the truth of the loop invariant ensures the truth of the post-condition of the loop.



#### Theorem 4.5.1 Loop Invariant Theorem

Let a **while** loop with guard  $G$  be given, together with pre- and post-conditions that are predicates in the algorithm variables. Also let a predicate  $I(n)$ , called the **loop invariant**, be given. If the following four properties are true, then the loop is correct with respect to its pre- and post-conditions:

- I. **Basis Property:** The pre-condition for the loop implies that  $I(0)$  is true before the first iteration of the loop
- II. **Inductive property:** For all integers  $k \geq 0$ , if the guard  $G$  and the loop invariant  $I(k)$  are both true before an iteration of the loop, then  $I(k + 1)$  is true after the iteration of the loop.
- III. **Eventual Falsity of Guard:** After a finite number of iterations of the loop, the guard  $G$  becomes false.
- IV. **Correctness of the Post-Condition:** If  $N$  is the least number of iterations after which  $G$  is false and  $I(N)$  is true, then the values of algorithm variables will be as specified in the post-condition of the loop.

## Chapter 5 Set Theory

### 5.1 Basic Definitions of a Set Theory

#### Subsets

**Definition** If  $A$  and  $B$  are sets, then  $A$  is called a **subset** of  $B$ , written  $A \subseteq B$ , if, and only if, every element of  $A$  is also an element of  $B$ .

Symbolically:

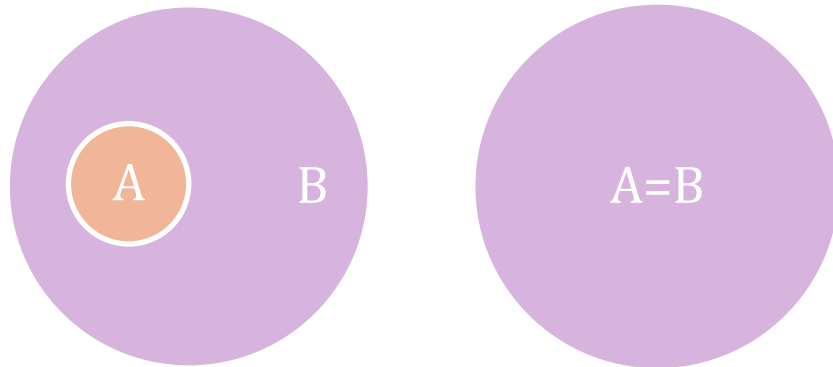
$$A \subseteq B \Leftrightarrow \forall x, \text{ if } x \in A \text{ then } x \in B$$

The phrases  $A$  is *contained in*  $B$  and  $B$  *contains*  $A$  are alternative ways of saying that  $A$  is a subset of  $B$ .

$$A \not\subseteq B \Leftrightarrow \exists x \text{ such that } x \in A \text{ and } x \notin B$$



**Definition** Let  $A$  and  $B$  be sets.  $A$  is a **proper subset** of  $B$  if, and only if, every element of  $A$  is in  $B$  but there is at least one element of  $B$  that is not in  $A$ .



#### Set Equality

**Definition** Given sets  $A$  and  $B$ ,  $A$  **equals**  $B$ , written  $A = B$ , if, and only if, every element of  $A$  is in  $B$  and every element of  $B$  is in  $A$ .

Symbolically:

$$A = B \Leftrightarrow A \subseteq B \text{ and } B \subseteq A.$$

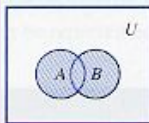
## Operations on Sets

### Definition

Let  $A$  and  $B$  be subsets of a universal set  $U$ .

The **union** of  $A$  and  $B$ , denoted  $A \cup B$ , is the set of all elements  $x$  in  $U$  such that  $x$  is in  $A$  or  $x$  is in  $B$ .

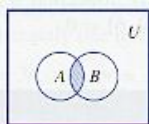
$$A \cup B = \{x \in U | x \in A \text{ or } x \in B\}$$



Shaded region represents  $A \cup B$ .

The **intersection** of  $A$  and  $B$ , denoted  $A \cap B$ , is the set of all elements  $x$  in  $U$  such that  $x$  is in  $A$  and  $B$ .

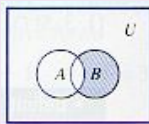
$$A \cap B = \{x \in U | x \in A \text{ and } x \in B\}$$



Shaded region represents  $A \cap B$ .

The **difference** of  $B$  minus  $A$  ( or **relative complement** of  $A$  in  $B$ ), denoted  $B - A$ , is the set of all elements  $x$  in  $U$  such that  $x$  is in  $B$  and  $x$  is not in  $A$ .

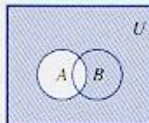
$$B - A = \{x \in U | x \in B \text{ and } x \notin A\}$$



Shaded region represents  $B - A$ .

The **complement** of  $A$ , denoted  $A^c$ , is the set of all elements  $x$  in  $U$  such that  $x$  is not in  $A$ .

$$A^c = \{x \in U | x \notin A\}$$



Shaded region represents  $A^c$ .

## The Empty Set

**Empty set or null set** – a set with no elements denoted  $\emptyset$

## Partitions of Sets

### Definition

Two sets are called **disjoint** if, and only if, they have no elements in common.

Symbolically:

$$A \text{ and } B \text{ are disjoint} \Leftrightarrow A \cap B = \emptyset$$

### Definition

Sets  $A_1, A_2, \dots, A_n$  are **mutually disjoint** (or **pair-wise disjoint** or **non-overlapping**) if, and only if, no two sets  $A_i$  and  $A_j$  with distinct subscripts have any element in common. More precisely, for all  $i, j = 1, 2, \dots, n$ ,

$$A_i \cap A_j \text{ whenever } i \neq j$$

### Definition

A collection of nonempty sets  $\{A_1, A_2, \dots, A_n\}$  is a **partition** of a set  $A$  if, and only if,

1.  $A = A_1 \cup A_2 \cup \dots \cup A_n$
2.  $A_1, A_2, \dots, A_n$  are mutually disjoint

### Power Sets

**Definition** Given as set  $A$ , the **power set** of  $A$ , denoted  $\mathcal{P}(A)$ , is the set of all subsets of  $A$

**Definition** Let  $n$  be a positive integer and let  $x_1, x_2, \dots, x_n$  be (not necessarily distinct) elements. The **ordered n-tuple**,  $(x_1, x_2, \dots, x_n)$ , consists of  $x_1, x_2, \dots, x_n$  together with ordering first  $x_1$ , then  $x_2$ , and so forth up to  $x_n$ . An ordered 2-tuple is called an **ordered pair**, and an ordered 3-tuple is called an **ordered triple**.

Two ordered  $n$ -tuples  $(x_1, x_2, \dots, x_n)$  and  $(y_1, y_2, \dots, y_n)$  are equal if, and only if,  $x_1 = y_1, x_2 = y_2, \dots, x_n = y_n$ .

Symbolically:

$$(x_1, x_2, \dots, x_n) = (y_1, y_2, \dots, y_n) \Leftrightarrow x_1 = y_1, x_2 = y_2, \dots, x_n = y_n$$

In particular,

$$(a, b) = (c, d) \Leftrightarrow a = c \text{ and } b = d$$

**Definition** Given two sets  $A$  and  $B$  the **Cartesian Product of A and B**, denoted  $A \times B$  (read A cross B), is the set of all ordered pairs  $(a, b)$  where  $a$  is in  $A$  and  $b$  is in  $B$ . Given sets  $A_1, A_2, \dots, A_n$ , the Cartesian product of  $A_1, A_2, \dots, A_n$  denoted  $A_1 \times A_2 \times \dots \times A_n$ , is the set of all ordered  $n$ -tuples  $(a_1, a_2, \dots, a_n)$  where  $a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n$ .

Symbolically:

$$A \times B = \{(a, b) | a \in A \text{ and } b \in B\}$$

$$A_1 \times A_2 \times \dots \times A_n = \{a_1, a_2, \dots, a_n | a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n\}$$

## An Algorithm to Check Whether One Set Is a Subset of Another (Optional)

### Algorithm 5.1.1 Testing Whether $A \subseteq B$

[Input sets  $A$  and  $B$  are represented as one-dimensional arrays  $a[1], a[2], \dots, a[n]$  and  $b[1], b[2], \dots, b[n]$ , respectively. Starting with  $a[1]$  and for each successive  $a[i]$  in  $A$ , a check is made to see whether  $a[i]$  is in  $B$ . To do this,  $a[i]$  is compared to successive elements of  $B$ . If  $a[i]$  is not equal to any element of  $B$ , then answer is given the value " $A \not\subseteq B$ ". If  $a[i]$  equals some element of  $B$ , the next successive element of  $A$  is found to be in  $B$ , then the answer never changes from its initial value " $A \subseteq B$ ".]

**Input:**  $m$  [a positive integer],  $a[1], a[2], \dots, a[m]$  [a one-dimensional array representing the set  $A$ ],  $n$  [a positive integer],  $b[1], b[2], \dots, b[m]$  [a one-dimensional array representing the set  $B$ ]

#### Algorithm Body:

```
i := 1, answer := "A ⊆ B"
while (j ≤ m and answer = "A ⊆ B")
    j := 1, found := no
    while (j ≤ n and found = "no")
        if a[i] = b[j] then found := "yes"
        j = j + 1
    end while
    [if found has not been given the value "yes" when execution
reaches this point, then a[i] ∈ B.]
    If found = "no" then answer := "A ⊄ B"
    i := i + 1
end while
```

**Output:** answer [a string]

## 5.2 Properties of Sets

### Theorem 5.2.1 Some Subset Relations

1. *Inclusion of Intersection:* For all sets  $A$  and  $B$ :  
 $A \cap B \subseteq A$  and  $A \cap B \subseteq B$
2. *Inclusion in Union:* For all sets  $A$  and  $B$ :  
 $A \subseteq A \cup B$  and  $B \subseteq A \cup B$
3. *Transitive Property of Subsets:* For all sets  $A$  and  $B$ :  
if  $A \subseteq B$  and  $B \subseteq C$ , then  $A \subseteq C$

### Element Argument: The Basic Method for Proving That One Set Is a Subset of Another

1. Let sets  $X$  and  $Y$  be given. To prove that  $X \subseteq Y$ ,
2. Suppose  $x$  is a particular but arbitrarily chosen element of  $X$
3. Show that  $x$  is an element of  $Y$

### Procedural Versions of Set Definitions

Let  $X$  and  $Y$  be subsets of a universal set  $U$  and suppose  $x$  and  $y$  are elements of  $U$ .

$$x \in X \cup Y \Leftrightarrow x \in X \text{ or } x \in Y$$

$$x \in X \cap Y \Leftrightarrow x \in X \text{ and } x \in Y$$

$$x \in X - Y \Leftrightarrow x \in X \text{ and } x \notin Y$$

$$x \in X^c \Leftrightarrow x \notin X$$

$$(x, y) \in X \times Y \Leftrightarrow x \in X \text{ or } y \in Y$$

### Set Identities

#### Theorem 5.2.2 Set Identities

Let all sets referred to below be subsets of a universal set  $U$

1. *Commutative Laws:* For all sets  $A$  and  $B$ ,

$$A \cup B = B \cup A \text{ and } A \cap B = B \cap A$$

2. *Associative Laws:* For all sets  $A$ ,  $B$ , and  $C$ ,

$$(A \cup B) \cup C = A \cup (B \cup C) \text{ and } (A \cap B) \cap C = A \cap (B \cap C)$$

3. *Distributive Laws:* For all sets  $A$ ,  $B$ , and  $C$ ,

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \text{ and } A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

4. *Identity Laws:* For all sets  $A$ ,

$$A \cup \emptyset = A \text{ and } A \cap U = A$$

5. *Complement Laws:*

$$A \cup A^c = U \text{ and } A \cap A^c = \emptyset$$

6. *Double Complement Law:* For all sets  $A$ ,

$$(A^c)^c = A$$

7. *Idempotent Laws:* For all sets  $A$ ,

$$A \cup A = A \text{ and } A \cap A = A$$

8. *Universal Bound Laws:* for all sets  $A$ ,

$$A \cup U = U \text{ and } A \cap \emptyset = \emptyset$$

9. *De Morgan's Laws:* For all sets  $A$  and  $B$ ,

$$(A \cup B)^c = A^c \cap B^c \text{ and } (A \cap B)^c = A^c \cup B^c$$

10. *Absorption Laws:* For all sets  $A$  and  $B$ ,

$$A \cup (A \cap B) = A \text{ and } A \cap (A \cup B) = A$$

11. *Complements of  $U$  and  $\emptyset$ :*

$$U^c = \emptyset \text{ and } \emptyset^c = U$$

12. *Set Difference Law:* For all sets  $A$  and  $B$ ,

$$A - B = A \cap B^c$$

### Proving Set Identities

#### Basic Method for Proving That Sets are Equal

Let sets  $X$  and  $Y$  be given. To prove  $X = Y$

1. Prove that  $X \subseteq Y$ .
2. Prove that  $Y \subseteq X$

#### Theorem 5.2.3 Intersection and Union with a Subset

For any sets  $A$  and  $B$ , if  $A \subseteq B$ , then

$$A \cap B = A \text{ and } A \cup B = B$$

### The Empty Set

#### Theorem 5.2.4 A Set with No Elements Is a Subset of Every Set

If  $E$  is a set with no elements and  $A$  is any set, then  $E \subseteq A$

#### Corollary 5.2.5 Uniqueness of the Empty Set

There is only one set, with no elements

#### Element Method for Proving a Set Equals the Empty Set

To prove that a set  $X$  is equal to the empty set  $\emptyset$ , prove that  $X$  has no elements. To do this suppose  $X$  has an element and derive a contradiction.

**Proposition 5.2.6** For all sets  $A$ ,  $B$ , and  $C$  if  $A \subseteq B$  and  $B \subseteq C^c$ , then  $A \cap C = \emptyset$

## 5.3 Disproofs, Algebraic Proofs, and Boolean Algebras

### Disproving an Alleged Set Property

Recall that to show a universal statement is false, it suffices to find one example (called a counterexample) for which it is false.

### The Number of Subsets of a Set

#### Theorem 5.3.1

For all integers  $n \geq 0$ , if a set  $X$  has  $n$  elements, then  $\wp(X)$  has  $2^n$  elements

### Boolean Algebras

**Definition** A **Boolean algebra** is a set  $B$  together with two operations, generally denoted  $+$  and  $\cdot$ , such that for all  $a$  and  $b$  in  $B$  both  $a + b$  and  $a \cdot b$  are in  $B$  and the following properties hold:

1. *Commutative Laws*: for all  $a$  and  $b$  in  $B$ ,  
$$a + b = b + a \text{ and } a \cdot b = b \cdot a$$
2. *Associative Laws*: For all  $a$ ,  $b$ , and  $c$  in  $B$ ,  
$$(a + b) + c = a + (b + c) \text{ and } (b)(a \cdot c) = a \cdot (b \cdot c)$$
3. *Distributive Laws*: For all  $a$ ,  $b$ , and  $c$  in  $B$ ,  
$$a + (b \cdot c) = (a + b) \cdot (a + c) \text{ and } a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$
4. *Identity Laws*: There exist distinct elements  $0$  and  $1$  in  $B$  such that for all  $a$  in  $B$ ,  
$$a + 0 = a \text{ and } a \cdot 1 = a$$
5. *Complement Laws*: For each  $a$  in  $B$ , there exists an element in  $B$ , denoted  $\bar{a}$  and called the **complement** or **negation** of  $a$ , such that  
$$a + \bar{a} = 1 \text{ and } a \cdot \bar{a} = 0$$

### Theorem 5.3.2 Properties of a Boolean Algebra

Let  $B$  be any Boolean Algebra.

1. *Uniqueness of the Complement Law:* For all  $a$  and  $x$  in  $B$ , if  $a + x = 1$  and  $a \cdot x = 0$  then  $x = \bar{a}$ .
2. *Uniqueness of 0 and 1:* If there exists  $x$  in  $B$  such that  $a + x = a$  for all  $a$  in  $B$ , then  $x = 0$ , and if there exists  $y$  in  $B$  such that  $a \cdot y = a$  for all  $a$  in  $B$ , then  $y = 1$ .
3. *Double Complement Law:* for all  $a \in B$ ,  $\overline{(\bar{a})} = a$ .
4. *Idempotent Law:* for all  $a \in B$ ,  
$$a + a = a \text{ and } a \cdot a = a$$
5. *Universal Bound Law:* For all  $a \in B$ ,  
$$a + 1 = a \text{ and } a \cdot 0 = 0$$
6. *De Morgan's Laws:* for all  $a$  and  $b \in B$ ,  
$$\overline{a + b} = \bar{a} \cdot \bar{b} \text{ and } \overline{a \cdot b} = \bar{a} + \bar{b}$$
7. *Absorption Laws:* For all  $a$  and  $b \in B$ ,  
$$(a + b) \cdot a = a \text{ and } (b)(a \cdot b) = a$$
8. *Complements of 0 and 1:*  
$$\bar{0} = 1 \text{ and } \bar{1} = 0$$

## 5.4 Russell's Paradox and the Halting Problem

**Russell's Paradox:** Most sets are not elements of themselves. However, we can imagine the possibility of a set being an element of itself. For example, the set of all abstract ideas might be considered an abstract idea. If we are allowed to use any description of a property as the defining property of a set, we can let  $S$  be the set of all sets that are not elements of themselves:

$$S = \{A \mid A \text{ is a set and } A \notin A\}$$

Is  $S$  and element of itself?

If  $S \in S$ , then  $S$  satisfies the defining property for  $S$ , and hence  $S \notin S$ . But if  $S$  is a set such that  $S \notin S$ , and so  $S$  satisfies the defining property for  $S$ , which implies that  $S \in S$ . Thus neither is  $S \in S$  nor is  $S \notin S$ , which is a contradiction. So  $S$  is an element of its self and it is not.

### The Halting Problem

#### Theorem 5.4.1

There is no computer algorithm that will accept any algorithm  $X$  and data set  $D$  as input and then will output, "halts" or "loops forever" to indicate whether  $X$  terminates in a finite number of steps when  $X$  is run with data set  $D$ .



## Chapter 6 Counting and Probability

---

### 6.1 Introduction

**Definition** A **sample space** is the set of all possible outcomes of a random process or experiment. An **event** is a subset of a sample.

#### Equally Likely Probability Formula

If  $S$  is a finite sample space in which all outcomes are equally likely and  $E$  is an event in  $S$ , then the probability of  $E$ , denoted  $P(E)$ , is

$$P(E) = \frac{\text{the number of outcomes in } E}{\text{the total number of outcomes in } S}$$

**Notation** for any finite set,  $N(A)$  denotes the number of elements in  $A$ .

$$P(E) = \frac{N(E)}{N(S)}$$

#### Counting the Elements of a List

##### Theorem 6.1.1 The Number of Elements in a List

If  $m$  and  $n$  are integers and  $m \leq n$ , then there are  $n - m + 1$  integers from  $m$  to  $n$  inclusive.

### 6.2 Possibility Trees and the Multiplication Rule

#### The Multiplication Rule

##### Theorem 6.2.1 The Multiplication Rule

If an operation consists of  $k$  steps and

The first step can be performed in  $n_1$  ways,

the second step can be performed in  $n_2$  ways [*regardless of how the first step was performed*],

⋮

the  $k$ th step can be performed in  $n_k$  ways [*regardless of how the preceding steps were performed*],

Then the entire operation can be performed in  $n_1 n_2 \cdots n_k$ .

**Definition** Let  $n$  be a positive integer. Given a finite set  $S$ , a **string of length  $n$  over  $S$**  is an ordered  $n$ -tuple of elements of  $S$  written without parentheses or commas. The elements of  $S$  are called **characters** of the string. The **null string** over  $S$  is defined to be “string” with no characters. It is usually denoted  $\epsilon$  and is said to have length 0. If  $S = \{1,0\}$ , then a string over  $S$  is called a **bit string**.

### Permutations

A permutation of a set of objects is an ordering of the objects in a row

#### Theorem 6.2.2

For any integer  $n$  with  $n \geq 1$ , the number of permutations of a set with  $n$  elements is  $n!$

### Permutations of Selected Elements

**Definition** An  $r$  – permutation of a set of  $n$  elements is an ordered selection of  $r$  elements taken from the set of  $n$  elements. The number of  $r$ -permutations of a set of  $n$  elements is denoted  $P(n, r)$

#### Theorem 6.2.3

If  $n$  and  $r$  are integers and  $1 \leq r \leq n$ , then the number of  $r$ -permutations of a set of  $n$  elements is given by the formula

$$P(n, r) = n(n - 1)(n - 2) \cdots (n - r + 1)$$

Or equivalently,

$$P(n, r) = \frac{n!}{(n - r)!}$$

## 6.3 Counting Elements of Disjoint Sets: The Addition Rule

#### Theorem 6.3.1 The Addition Rule

Suppose a finite set  $A$  equals the union of  $k$  distinct mutually disjoint subsets  $A_1, A_2, \dots, A_k$ . Then

$$N(A) = N(A_1) + N(A_2) + \cdots + N(A_k)$$

### The Difference Rule

#### Theorem 6.3.2 The Difference Rule

If  $A$  is a finite set and  $B$  is a subset of  $A$ , then

$$N(A - B) = N(A) - N(B)$$

#### Formula for the Probability of the Complement of an Event

If  $S$  is a finite sample space and  $A$  is an event in  $S$ , then

$$P(A^c) = 1 - P(A)$$

### The Inclusion/Exclusion Rule

#### Theorem 6.3.3 The Inclusion Exclusion Rule for Two or Three Sets

If  $A$ ,  $B$ , and  $C$  are any finite sets, then

$$N(A \cup B) = N(A) + N(B) - N(A \cap B)$$

And

$$N(A \cup B \cup C) = N(A) + N(B) + N(C) + N(A \cap B) - N(A \cap C) - N(B \cap C) - N(A \cap B \cap C)$$

## 6.4 Counting Subsets of a Set: Combinations

**Definition** Let  $n$  and  $r$  be nonnegative integers with  $r \leq n$ . An  **$r$ -combination** of set  $n$  elements is a subset of  $r$  of the  $n$  elements. The symbol  $\binom{n}{r}$ , which is “ $n$  choose  $r$ ” denotes the number of subsets of size  $r$  ( $r$ -combinations) that can be chosen from a set of  $n$  elements.

**Theorem 6.4.1** The number of subsets of size  $r$  (or  $r$ -combinations) that can be chosen from a set of  $n$  elements,  $\binom{n}{r}$ , is given by the formula

$$\binom{n}{r} = \frac{P(n, r)}{r!}$$

Or equivalently,

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}$$

Where  $n$  and  $r$  are nonnegative integers with  $r \leq n$ .

**Theorem 6.4.2** Suppose a collection consists of  $n$  objects of which  
 $n_1$  are of type 1 and are indistinguishable from each other  
 $n_2$  are of type 2 and are indistinguishable from each other  
 $\vdots$   
 $n_k$  are of type  $k$  and are indistinguishable from each other,

And suppose that  $n_1 + n_2 + \dots + n_k = n$ . Then the number of distinct permutations of the  $n$  objects is

$$\binom{n}{n_1} \binom{n-n_1}{n_2} \binom{n-n_1-n_2}{n_3} \dots \binom{n-n_1-n_2-\dots-n_{k-1}}{n_k} = \frac{n!}{n_1! n_2! n_3! \dots n_k!}$$

## 6.5 $r$ -Combinations with Repetition Allowed

**Definition** An  **$r$ -combination with repetition allowed**, or **multiset of size  $r$** , chosen from a set  $X$  of  $n$  elements is an unordered selection of elements taken from  $X$  with repetition allowed. If  $X = \{x_1, x_2, \dots, x_n\}$ , we write an  $r$ -combination repetition allowed, or a multiset of size  $r$ , as  $[x_{i_1}, x_{i_2}, \dots, x_{i_r}]$  where each  $x_{i_j}$  is in  $X$  and some of the  $x_{i_j}$  may equal each other.

**Theorem 6.5.1** The number of  $r$ -combinations with repetition allowed (multisets of size  $r$ ) that can be selected from a set of  $n$  elements is

$$\binom{r+n-1}{r}$$

This equals the number of ways  $r$  objects can be selected from  $n$  categories of objects with repetition allowed.

## 6.6 The Algebra of Combinations

### Pascal's Formula

**Theorem 6.6.1 Pascal's Formula** Let  $n$  and  $r$  be positive integers and suppose  $r \leq n$ . Then

$$\binom{n+1}{r} = \binom{n}{r-1} + \binom{n}{r}$$

## 6.8 Probability Axioms and Expected Value

**Probability Axioms** Let  $S$  be a sample space, and let  $A$  and  $B$  be any events in  $S$ . Then

$$0 \leq P(A) \leq 1$$

$$P(\emptyset) = 0 \text{ and } P(S) = 1$$

If  $A$  and  $B$  are disjoint (that is, if  $A \cap B = \emptyset$ ), then the probability of the union of  $A$  and  $B$  is

$$P(A \cup B) = P(A) + P(B)$$

**Probability of the Complement of an Event** If  $A$  is any event in a sample space  $S$ , then

$$P(A^c) = 1 - P(A)$$

**Probability of a General Union of Two Events** If  $S$  is any sample space and  $A$  and  $B$  are any elements in  $S$ , then

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

### Expected Value

**Definition** Suppose the possible outcomes of an experiment or random process, are real numbers  $a_1, a_2, a_3, \dots, a_n$  which occur with probabilities  $p_1, p_2, p_3, \dots, p_n$ . The **expected value** of the process is

$$\sum_{k=1}^n a_k p_k = a_1 p_1 + a_2 p_2 + a_3 p_3 + \dots + a_n p_n$$

## 6.9 Conditional Probability, Bayer's Formula, and Independent Events

### Conditional Probability

**Definition** Let  $A$  and  $B$  be events in a sample space  $S$ . If  $P(A) \neq 0$ , then the **conditional probability of  $B$  given  $A$** , denoted  $P(B|A)$ , is

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

### Bayes' Theorem

#### Theorem 6.9.1 Bayes' Theorem

Suppose that a sample space  $S$  is a union of mutually disjoint events  $B_1, B_2, B_3, \dots, B_n$ . Suppose  $A$  is an event in  $S$ , and suppose  $A$  and all the  $B_i$  have nonzero probabilities. If  $k$  is an integer with  $1 \leq k \leq n$ , then

$$P(B_k|A) = \frac{P(A|B_k) \cdot P(B_k)}{P(A|B_1) \cdot P(B_1) + P(A|B_2) \cdot P(B_2) + \dots + P(A|B_n) \cdot P(B_n)}$$

### Independent Events

**Definition** If  $A$  and  $B$  are events in a sample space  $S$ , then  $A$  and  $B$  are **independent** if, and only if,

$$P(A \cap B) = P(A) \cdot P(B)$$

**Definition** Let  $A$ ,  $B$ , and  $C$  be events in sample space  $S$ .  $A$ ,  $B$ , and  $C$  are **pairwise independent** if, and only if, they satisfy the following conditions

$$P(A \cap B) = P(A) \cdot P(B)$$

$$P(A \cap C) = P(A) \cdot P(C)$$

$$P(B \cap C) = P(B) \cdot P(C)$$

$A$ ,  $B$ , and  $C$  are **mutually independent** if, and only if, they satisfy the following conditions

$$P(A \cap B) = P(A) \cdot P(B)$$

$$P(A \cap C) = P(A) \cdot P(C)$$

$$P(B \cap C) = P(B) \cdot P(C)$$

$$P(A \cap B) = P(A) \cdot P(B)$$

$$P(A \cap C) = P(A) \cdot P(C)$$

$$P(A \cap B \cap C) = P(A) \cdot P(B) \cdot P(C)$$

## Chapter 7 Functions

### 7.1 Functions Defined on General Sets

**Definition** A function  $f$  from a set  $X$  to a set  $Y$  is a relation between elements of  $X$ , called **inputs**, and elements of  $Y$ , called **outputs**, with the property that each input is related to one and only one output. The notation  $f: X \rightarrow Y$  means that  $f$  is a function from  $X$  to  $Y$ .  $X$  is called the **domain** of  $f$ , and  $Y$  is called the **codomain** of  $f$ . Given an input element in  $X$ , there is a unique output element in  $Y$  that is related to  $x$  by  $f$ . We say that “ $f$  sends  $x$  to  $y$ ” and write  $x \xrightarrow{f} y$  or  $f: x \rightarrow y$ . The unique element  $y$  which  $f$  sends  $x$  is denoted  $f(x)$  and is called  $f$  of  $x$  or:

The output of  $f$  for the input  $x$

The value of  $f$  at  $x$

The image of  $x$  under  $f$

The set of all values of  $f$  taken together is called the range of  $f$  or the image of  $X$  under  $f$ . Symbolically

$$\text{range of } f = \text{image of } X \text{ under } f = \{y \in Y \mid y = f(x), \text{ for some } x \in X\}$$

Given an element  $y$  in  $Y$ , there may exist elements in  $X$  with  $y$  as their image. If  $f(x) = y$ , then  $x$  is called a **preimage of  $y$**  or an **inverse image of  $y$** . The set of all inverse images of  $y$  is called the **inverse image of  $y$** . Symbolically,

$$\text{inverse image of } y = \{x \in X \mid f(x) = y\}$$

#### Arrow Diagrams

If  $X$  and  $Y$  are finite sets, you can define a function  $f$  from  $X$  to  $Y$  by making a list of elements in  $X$  and a list of elements in  $Y$  and drawing an arrow from each element of  $X$  to the corresponding element in  $Y$ . Such a drawing is called an arrow diagram. The definition of function implies that the arrow diagram for a function  $f$  has the following two properties:

1. Every element of  $X$  has an arrow coming out of it
2. No element of  $X$  has two arrows coming out of it that point to two different elements of  $Y$ .

#### Function Machines

Another useful way to think of a function is as a machine. Suppose  $f$  is a function from  $X$  to  $Y$  and an input of  $x$  is given. Imagine  $f$  to be a machine that processes  $x$  in a certain way to produce the output  $f(x)$ .

**Definition** Suppose  $f$  and  $g$  are functions from  $X$  to  $Y$ . Then  $f$  equals  $g$ , written  $f = g$  if, and only if,

$$f(x) = g(x) \quad \text{for all } x \in X$$

#### Boolean Functions

**Definition** An ( $n$ -place) **Boolean Function**  $f$  is a function whose domain is the set of all ordered  $n$ -tuples of 0's and 1's and whose codomain is the set  $\{0,1\}$ . More formally, the domain of a Boolean function can be described as the Cartesian product of  $n$  copies of the set  $\{0,1\}$ , which is denoted  $\{0,1\}^n$ , thus  $f: \{0,1\}^n \rightarrow \{0,1\}$ .

### Checking Whether a Function is Well Defined

A function is **not well defined** if the formula does not define a function

## 7.2 One-to-One and Onto, Inverse Functions

### One-to-One Functions

**Definition** Let  $F$  be a function from a set  $X$  to a set  $Y$ .  $F$  is **one-to-one** (or **injective**) if, and only if, for all elements  $x_1$  and  $x_2$  in  $X$ ,

$$\text{if } F(x_1) = F(x_2), \text{ then } x_1 = x_2$$

Or equivalently,

$$\text{if } x_1 \neq x_2, \text{ then } F(x_1) \neq F(x_2)$$

Symbolically

$$F: X \rightarrow Y \text{ is one-to-one} \Leftrightarrow \forall x_1, x_2 \in X, \text{ if } F(x_1) = F(x_2) \text{ then } x_1 = x_2$$

A function  $F: X \rightarrow Y$  is not one-to-one  $\Leftrightarrow \exists$  elements  $x_1, x_2$  in  $X$ , with  $F(x_1) = F(x_2)$  and  $x_1 \neq x_2$

### Onto Functions

**Definition** Let  $F$  be a function from a set  $X$  to a set  $Y$ .  $F$  is **onto** (or **surjective**) if, and only if, given an element  $y$  in  $Y$ , it is possible to find an element  $x$  in  $X$  with property that  $y = F(x)$ .

Symbolically:

$$F: X \rightarrow Y \text{ is onto} \Leftrightarrow \forall y \in Y, \exists x \in X \text{ such that } F(x) = y$$

$$F: X \rightarrow Y \text{ is not onto} \Leftrightarrow \exists y \in Y \text{ such that } \forall x \in X, F(x) \neq y$$

### One-to-one Correspondences

**Definition** a **one-to-one correspondence** (or **bijection**) from a set  $X$  to a set  $Y$  is a function  $F: X \rightarrow Y$  that is both one-to-one and onto

### Inverse Functions

**Theorem 7.2.1** Suppose  $F: X \rightarrow Y$  is a one-to-one correspondence; that is, suppose  $F$  is one-to-one and onto. Then there is a function  $F^{-1}: Y \rightarrow X$  that is defined as follows:

Given any element  $y$  in  $Y$ ,

$$F^{-1}(y) = \text{that unique element } x \text{ in } X \text{ such that } F(x) = y$$

In other words,

$$F^{-1}(y) = x \Leftrightarrow y = f(x)$$

**Definition** The function  $F^{-1}$  of theorem 7.2.1 is called the **inverse function** of  $F$ .

**Theorem 7.2.2** If  $X$  and  $Y$  are sets and  $F: X \rightarrow Y$  is one-to-one and onto, then  $F^{-1}: Y \rightarrow X$  is also on-to-one and onto

### 7.3 Application: The Pigeonhole Principle

**Pigeonhole Principle** A function from one finite set to a smaller finite set cannot be one-to-one: there must be at least two elements in the domain that have the same image in the co-domain.

#### Generalized Pigeonhole Principle

**Generalized Pigeonhole Principle** for any function from a finite set  $X$  to a finite set  $Y$  and for any positive integer  $k$  if  $N(X) > k \cdot N(Y)$ , then there is some  $y \in Y$  such that  $y$  is the image of at least  $k + 1$  distinct elements of  $X$ .

**Generalized Pigeonhole Principle (Contrapositive Form)** for any function  $f$  from a finite set  $X$  to a finite set  $Y$  and for any positive integer  $k$ , if for each  $y \in Y$ ,  $f^{-1}(y)$  has at most  $k$  elements, then  $X$  has at most  $k \cdot N(Y)$  elements.

#### Proof of the Pigeonhole Principle

**Definition** a set is called **finite** if, and only if, it the empty set or there is a one-to-one correspondence from  $\{1, 2, \dots, n\}$  to it, where  $n$  is a positive integer. In the first case the **number of elements** in the set is said to be 0, and in the second case it is said to be  $n$ . A set that is not finite is called **infinite**.

### 7.4 Composition of Functions

**Definition** Let  $f: X \rightarrow Y'$  and  $g: Y' \rightarrow Z$  be functions with the property that the range of  $f$  is a subset of the domain of  $g$ . Define a new function  $g \circ f: X \rightarrow Z$  as follows:

$$(g \circ f)(x) = g(f(x)) \text{ for all } x \in X$$

Where  $g \circ f$  is read “ $g$  circle  $f$ ” and  $g(f(x))$  is read “ $g$  of  $f$  of  $x$ .” The function  $g \circ f$  is called the **composition of  $f$  and  $g$** .

#### Theorem 7.4.1 Composition with an Identity Function

If  $f$  is a function from a set  $X$  to a set  $Y$ , and  $i_x$  is the identity function on  $X$ , and  $i_y$  is the identity function on  $Y$ , then

$$f \circ i_x = f \text{ and } i_y \circ f = f$$

#### Theorem 7.4.2 Composition of a Function with its Inverse

If  $f: X \rightarrow Y$  is a one-to-one and onto function with inverse function  $F^{-1}: Y \rightarrow X$ , then

$$f^{-1} \circ f = i_x \text{ and } f \circ f^{-1} = i_y$$



**Theorem 7.4.3**

If  $f: X \rightarrow Y$  and  $g: Y \rightarrow Z$  are both one-to-one functions then  $g \circ f$  is one-to-one.

***Composition of Onto Functions***

**Theorem 7.4.4**

If  $f: X \rightarrow Y$  and  $g: Y \rightarrow Z$  are both onto functions, then  $g \circ f$  is onto.

## Chapter 8 Recursion

---

### 8.1 Recursively Defined Sequences

**Definition** A **recurrence relation** for a sequence  $a_0, a_1, a_2, \dots$  is a formula that relates each term  $a_k$  to a certain of its predecessors  $a_{k-1}, a_{k-2}, \dots, a_{k-i}$ , where  $i$  is an integer and  $k$  is any integer greater than or equal to  $i$ . The **initial conditions** for such a recurrence relation specify the values of  $a_0, a_1, a_2, \dots, a_{i-1}$ , if  $i$  is a fixed integer, or  $a_0, a_1, a_2, \dots, a_m$ , where  $m$  is an integer with  $m \geq 0$ , if  $i$  depends on  $k$ .

### 8.2 Solving Recurrence Relations by Iteration

Suppose you have a sequence that satisfies a certain recurrence relation and initial conditions. It is often helpful to know an explicit formula for the sequence, especially if you need to compute terms with very large subscripts or if you need to examine general properties of the sequence. Such an explicit formula is called a **solution** to the recurrence relation.

#### *The Method of Iteration*

The most basic method for finding an explicit formula for a recursively defined sequence is iteration. Iteration works as follows: given a sequence  $a_0, a_1, a_2, \dots$  defined by a recurrence relation and initial conditions, you start from the initial conditions and calculate successive terms of the sequence until you see a pattern developing. At that point you guess an explicit formula.

**Definition** A sequence  $a_0, a_1, a_2, \dots$  is called an **arithmetic sequence** if, and only if, there is a constant  $d$  such that

$$a_k = a_{k-1} + d$$

Or equivalently,

$$a_n = a_0 + dn$$

**Definition** A sequence  $a_0, a_1, a_2, \dots$  is called a **geometric sequence** if, and only if, there is a constant  $r$  such that

$$a_k = ra_{k-1} \text{ for all integers } k \geq 1$$

Or, equivalently,

$$a_n = a_0 r^n \text{ for all integers } k \geq 0$$

#### *Checking the Correctness of a Formula by Mathematical Induction*

Use mathematical induction to prove the function is correct