Discrete Structures

By:
Duy Bui

Discrete Structures

By:
Duy Bui

Online:
< http://cnx.org/content/col10513/1.1/ >

C O N N E X I O N S

Rice University, Houston, Texas

Collection structure revised: January 23, 2008
PDF generated: October 26, 2012
For copyright and attribution information for the modules contained in this collection, see p. 109.

Table of Contents

Chapter 1

Discrete Structures Course Information[1]

Vietnam National University, Hanoi
College of Technology
CS281 - Discrete Structures
Spring, 2008
Student Manual

1.1 Letter to Student

To the Student:

This course and this Student Manual reflect a collective effort led by your instructor. This course is an important component of our academic program. Although it has been offered for many years, this latest version represents an attempt to expand the range of sources of information and instruction, so that the course continues to be up-to-date and the methods well suited to what is to be learned.

You will be asked from time to time to offer feedback on how the Student Manual is working and how the course is progressing. Your comments will inform the development team about what is working, and what requires attention. Our goal is to help you learn what is important about this particular field, and to eventually succeed as a professional applying what you learn in this course.

This Student Manual is designed to assist you through the course by providing specific information about student responsibilities including requirements, timelines and evaluations.

I hope you enjoy the course.

1.2 Faculty Information

Name: Bui The Duy Office Location: 306, E3 Building
Email: duybt@vnu.edu.vn
Office Hours: 8am-5pm, weekdays
Before or after class: 10am-11am, Tuesday
Support personnel:
- Le Thi Hoi – Assistant, 306, E3 Building
- Ngo Thi Duyen – Assistant, 306, E3 Building
- Ma Thi Chau – Assistant, 306, E3 Building

[1] This content is available online at <http://cnx.org/content/m14585/1.3/>.

1.3 Resources

- Course Reading material
- MIT's OpenCourseWare[2]
- Connexions[3]
- On-line Discrete Math tutorials [4]

1.4 Purpose of the Course

The main goal of this course is to provide students with an opportunity to gain an understanding of the theoretical foundations of Computer Science. The main areas of the course are Mathematical Logic, Set Theory, and Relations. Topics include proof methods with emphasis on mathematical induction, solving recurrence relations, propositional logic, first order logic, proof techniques, mathematical induction, sets, operations on sets, relations, operations on relations, and functions. The emphasis is on the applications of discrete structures in computer science rather than the mathematical theory itself.

1.5 Course Description

Discrete structures is foundational material for computer science. By foundational we mean that relatively few computer scientists will be working primarily on discrete structures, but that many other areas of computer science require the ability to work with concepts from discrete structures. Discrete structures includes important material from such areas as set theory, logic, graph theory, and combinatorics.

This course covers the mathematics that underlies most of computer science, which are the fundamental mathematical concepts and reasoning along with problem solving techniques. Topics covered include propositional logic, predicate logic, inferencing, proof methods including induction, set operations, binary relations including order relations, and equivalence relations, graphs, and functions.

1.6 Course Requirements

- CS101 - Introduction to Programming course
- MATH102 - Pre-Calculus II, or equivalents.
- Calculus is preferred, but not required.

Students are presumed to be familiar with basic programming techniques, including the use of functions and procedures, loops and recursion. Also assumed is facility with basic algebra.

Students are also expected to be familiar with the use of standard Internet-based tools including e-mail.

1.7 Course Objectives

- Cultivate clear thinking and creative problem solving.
- Thoroughly train in the construction and understanding of mathematical proofs. Exercise common mathematical arguments and proof strategies.
- Cultivate a sense of familiarity and ease in working with mathematical notation and common concepts in discrete mathematics.
- Teach the basic results in Mathematical Logic, Set Theory, and Relations.

[2] http://ocw.mit.edu/index.html
[3] http://cnx.org/
[4] http://math.about.com/cs/discretemath/

1.8 Student Objectives

At the end of the course, students should:

- Understand fundamental mathematical concepts as they apply to computer science by seeing how mathematics supports CS, and how CS concepts can be formalized in mathematics
- Illustrate by examples the basic terminology of functions, relations, and sets and demonstrate knowledge of their associated operations.
- Establish and solve recurrence relations that arise in counting problems including the problem of determining the time complexity of recursively defined algorithms.
- Model logic statements arising in algorithm correctness and real-life situations and manipulate them using the formal methods of propositional and predicate logic.
- Outline basic proofs for theorems using the techniques of: direct proofs, proof by counterexample, proof by contraposition, proof by contradiction, and mathematical induction.
- Relate the ideas of mathematical induction to recursion and recursively defined structures.
- Enhance one's ability to program by seeing how mathematical concepts form the basis for many common programming problems (e.g. grammars for parsing, predicate calculus for logic programming, sets and algebras for relational databases, graphs and topological sorting for automating optimization).
- Further their ability to write large programs by integrating code from a diverse spectrum of program components.

1.9 Grading Procedures

The overall grade for this course is based on your performance in (i) exercises, (ii) assignments, (iii) mid-term exam and (iv) final exam, with weights as given below. Exams consist of a midterm and a final exam.

Course component grading weight (it can be changed):

- Midterm: 20%
- Weekly homework: 40%
- Final: 40%

1.10 Content Information

First we learn a general methodology for solving problems. This methodology is going to be followed in solving problems, and in proving theorems throughout this course.

The next subject is logic. Logic subject matter is covered in Chapter 1 of the textbook. "Logic" is a language that captures the essence of our reasoning, and correct reasoning must follow the rules of this language. We start with logic of sentences called propositional logic, and study elements of logic, (logical) relationships between propositions, and reasoning. Then we learn a little more powerful logic called predicate logic. Predicate logic allows us to reason with statements involving variables among other statements.

In Chapter 1, we also study sets, relations between sets, and operations on sets. Sets are the basis of every theory in computer science and mathematics.

In Chapter 3, we learn recursive definitions and mathematical reasoning, in particular induction. There are sets, operations and functions that can be defined precisely by recursive definitions. Properties of those recursively defined objects can be established rigorously using proof by induction.

Then in Chapters 6 we study relations. Relations are one of the key concepts in the discussion of many subjects on computer and computation. For example, a database is viewed as a set of relations and database query languages are constructed based on operations on relations and sets. Graphs are also covered briefly here. Graphs are an example of discrete structures and they are one of the most useful models for computer scientists and engineers in solving problems. More in-depth coverage of graphs can be found in Chapter 7.

Finally, back in Chapter 1 again, we briefly study functions. Functions are a special type of relation and basically the same kind of concept as the ones we see in calculus. However, functions are one of the most important concepts in the discussion of many subjects on computer and computation, such as data structures, database, formal languages and automata, and analysis of algorithms, which is briefly covered in Chapter 2.

1.11 Instructional Sequence

Unit 1

Task 1: Read the following:

- Introduction to Discrete Structures
- Problem Solving Framework
- Problem Solving Example 1

Unit 2

Task 1: Read the following:

- Problem Solving Examples

Unit 3

Task 1: Read the following:

- Introduction to Logic
- What is Proposition
- Elements of Propositional Logic
- Truth Table
- Connectives
- Construction of Proposition
- Converse and Contrapositive

These materials can also be found in Textbook 1.1 - 1.2.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 11 : 1 all
- Textbook p. 11 : 3 all
- Textbook p. 11 : 7 a c e g
- Textbook p. 12 : 9 b d f h
- Textbook p. 13 : 19 all
- Textbook p. 13 : 21 a c e
- Textbook p. 13 : 23 d f
- Reading Material: Chapter Logic - Exercise 16-21

Unit 4

Task 1: Read the following:

- Variations of if_then
- From English to Proposition

These materials can also be found in Textbook 1.1 - 1.2.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 12 : 15 all
- Textbook p. 12 : 17 all
- Reading Material: Chapter Logic - Exercise 22-23

Unit 5

Task 1: Read the following:

- Introduction to Reasoning
- Identities of Propositions and Dual
- Example of Use of Identities

These materials can also be found in Textbook 1.1 - 1.2.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 19 : 1 a d f
- Textbook p. 19 : 5
- Textbook p. 20: 9 b d f
- Textbook p. 20: 11 a
- Textbook p. 20: 20 all
- Textbook p. 20: 25
- Reading Material: Chapter Logic - Exercise 24-29

Unit 6

Task 1: Read the following:

- Implications
- Reasoning with Propositions
- Proof of Identities
- Proof of Implications

These materials can also be found in Textbook 1.1 - 1.2 and pp. 167 - 173.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Converting Inferencing to Logic
- Check the Correctness of Reasoning of 1. above using Inference Check
- Textbook p. 183: 1 all
- Textbook p. 183: 3
- Textbook p. 20 : 9 c d e
- Textbook p. 20 : 19
- Reading Material: Chapter Logic - Exercise 30-31

Unit 7

Task 1: Read the following:

- Why Predicate Logic ?
- Predicate
- Quantification
- Constructing Formulas (Wffs)

These materials can also be found in Textbook 1.3.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 33 : 3 all
- Textbook p. 33 : 5 all
- Textbook p. 35 : 19 all
- Textbook p. 36 : 23 a c e
- Reading Material: Chapter Logic - Exercise 32-35

Unit 8

Task 1: Read the following:

- From Wff to Proposition
- English to Logic Translation

These materials can also be found in Textbook 1.3.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 34 : 13 b d f h
- Textbook p. 35 : 17 a c e g
- Textbook p. 36 : 31 all
- Textbook p. 36 : 33 all
- Converting English to Logic
- Reading Material: Chapter Logic - Exercise 36-39

Unit 9

Task 1: Read the following:

- Reasoning with Predicate Logic

These materials can also be found in Textbook 1.3 and 3.1 .

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 37 : 35
- Textbook p. 183: 9 all
- Textbook p. 183: 11 b d
- Reading Material: Chapter Logic - Exercise 40-42

Unit 10

Task 1: Read the following:

- Quantifiers and Connectives

These materials can also be found in Textbook 1.3.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 37 : 41
- Textbook p. 37 : 43 a
- Reading Material: Chapter Logic - Exercise 43-44

Unit 11

Task 1: Read the following:

- Introduction to Sets
- Representation of Set

- Equality, Subset, etc.

These materials can also be found in Textbook 1.4.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 45 : 1 a d
- Textbook p. 45 : 3 all
- Textbook p. 45 : 5 all
- Textbook p. 45 : 9
- Textbook p. 45 : 13 all
- Reading Material: Chapter Set Theory - Exercise 4-8

Unit 12

Task 1: Read the following:

- Mathematical Reasoning
- Set Operations

These materials can also be found in Textbook 1.3 and 1.5.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 45 : 21
- Textbook p. 45 : 23
- Textbook p. 54 : 3 all
- Textbook p. 54 : 19 a c
- Textbook p. 54 : 21
- Reading Material: Chapter Set Theory - Exercise 9-13

Unit 13

Task 1: Read the following:

- Properties of Set Operation

These materials can also be found in Textbook 1.5.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 54 : 7 all
- Textbook p. 54 : 9 a
- Textbook p. 54 : 15 a
- Reading Material: Chapter Set Theory - Exercise 14-16

TEST: Covers Unit 3 - Unit 12 inclusive. Unit 14

Task 1: Read the following:

- Recursive Definition
- Generalized Set Operations

These materials can also be found in Textbook 1.5 and 3.3.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 55 : 35 a

- Textbook p. 55 : 36 b
- Textbook p. 210: 21
- Textbook p. 210: 23 all
- Textbook p. 210: 31 (An empty string is a string with no symbols in it.)
- Reading Material: Chapter Recursion - Exercise 5-9

Unit 15

Task 1: Read the following:

- Recursive Definition of Function
- Recursive Algorithm

These materials can also be found in Textbook 3.3 and 3.4.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 209: 1 a b c
- Textbook p. 209: 3 a b
- Textbook p. 209: 7
- Textbook p. 218: 1
- Textbook p. 218: 3
- Reading Material: Chapter Recursion - Exercise 10-14

Unit 16

Task 1: Read the following:

- First Principle of Mathematical Induction

These materials can also be found in Textbook 3.2.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 199: 3
- Textbook p. 200: 9
- Textbook p. 200: 13
- Textbook p. 200: 19
- Textbook p. 200: 21
- Textbook p. 201: 43
- Reading Material: Chapter Recursion - Exercise 15-20

Unit 17

- Task 1: Read the following:
  - Example of Use of Induction
  - Second Principle of Mathematical Induction
- Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.
  - Textbook p. 199: 5
  - Textbook p. 202: 59
  - Reading Material: Chapter Recursion - Exercise 21-22

These materials can also be found in Textbook 3.2.

Unit 18

Task 1: Read the following:

- Introduction to Relation
- Binary Relation
- Definition of Relation (general relation)
- Equality of Relations
- Recursive Definition of Relation

These materials can also be found in Textbook 6.1 and 6.2.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 382: 1 all
- Textbook p. 382: 2 a
- Textbook p. 389: 3
- Recursively define the relation $\{< a, b > \mid a = 2b \}$.
- List unary relation on { 1, 2, 3 }.
- Prove that there are 2n2 binary relations on a set of cardinality n.
- Reading Material: Chapter Relation - Exercise 10-13

Unit 19

Task 1: Read the following:

- Digraph
- Digraph Representation of Binary Relation
- Properties of Binary Relation

These materials can also be found in Textbook 6.3, 7.1 and 7.2.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 382: 3 b d f
- Textbook p. 382: 5 a c g e
- Textbook p. 383: 19 a b
- Textbook p. 396: 12
- Textbook p. 396: 13
- Textbook p. 396: 15
- Reading Material: Chapter Relation - Exercise 14-17

Unit 20

Task 1: Read the following:

- Operations on Binary Relations
- Closures of Binary Relation

These materials can also be found in Textbook 6.1 and 6.4.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 383: 19 a b
- Textbook p. 383: 21
- Textbook p. 383: 35
- Textbook p. 406: 1
- Textbook p. 406: 3
- Textbook p. 406: 11 for 5
- Textbook p. 407: 22

- Reading Material: Chapter Relation - Exercise 18-22

Unit 21

Task 1: Read the following:

- Equivalence Relation
- Order Relation (Partial, Total, and Quasi Orders)

These materials can also be found in Textbook 6.5 and 6.6.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 413: 1 a c e
- Textbook p. 413: 5 a b
- Textbook p. 413: 9
- Textbook p. 413: 11
- Textbook p. 414: 23
- Textbook p. 414: 25
- Textbook p. 414: 31 a b
- Reading Material: Chapter Relation - Exercise 23-28

Unit 22

Task 1: Read the following:

- Order Relation (Minimal Element and the rest)

These materials can also be found in Textbook 6.6.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 428: 1
- Textbook p. 428: 3
- Textbook p. 428: 5
- Textbook p. 428: 15 a d
- Textbook p. 428: 17
- Textbook p. 429: 27
- Reading Material: Chapter Relation - Exercise 29-31

Unit 23

Task 1: Read the following:

- Definitions on Function
- Growth of Functions

These materials can also be found in Textbook 1.6 and 1.8.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 67: 1
- Textbook p. 67: 5 a c
- Textbook p. 67: 10 a b c
- Textbook p. 67: 11 a b c
- Textbook p. 67: 15 a b
- Textbook p. 68: 17 a c

- Textbook p. 68: 49
- Textbook p. 90: 1
- Textbook p. 90: 3
- Reading Material: Chapter Function - Exercise 3-9

Unit 24

Task 1: Read the following:

- Growth of Functions (Calculation of Big-Oh Relation)

These materials can also be found in Textbook 1.8.

Task 2: Do the following exercises: These exercises are NOT homework questions. They are for helping you understand the materials of this unit.

- Textbook p. 90: 5
- Textbook p. 90: 11
- Textbook p. 90: 13
- Textbook p. 90: 15
- Textbook p. 91: 19 a b
- Textbook p. 91: 31
- Reading Material: Chapter Function - Exercise 10-14

1.12 Calendar – Timetable

Week	Units to Study
1	Unit 1, Unit 2
2	Unit 3, Unit 4
Submit Homeworks 1, 2	
3	Unit 5, Unit 6
4	Unit 7, Unit 8
Submit Homeworks 3, 4	
5	Unit 9, Unit 10
6	Unit 11, Unit 12
Submit Homeworks 5, 6	
7	Unit 13, Unit 14
TEST : Unit 3 - Unit 12 inclusive	
8	Unit 15, Unit 16
Submit Homeworks 7, 8	
9	Unit 17, Unit 18
10	Unit 19, Unit 20
Submit Homeworks 9, 10	
11	Unit 21, Unit 22
12	Unit 23, Unit 24
Submit Homeworks 11, 12	
EXAM : Unit 3 - Unit 24 inclusive	

Table 1.1

1.13 Readings

- Course Reading Material
- Textbook: Kenneth H. Rosen, Discrete Mathematics and Its Applications, 6th edition, McGraw-Hill Science/Engineering/Math, 2006, ISBN 978-0073312712.

1.14 Reference

- Task Force on Computing Curricula. Computing Curricula 2001: Computer Science, Final Report, December 2001. Available at http://www.sigcse.org/cc2001/

- Discrete Mathematical Structures, 5th edition, by B. Kolman, R.C. Busby, and S.C. Ross, published by Prentice Hall, 2004.
- Mathematical Structures for Computer Science, 5th edition, by J.L. Gersting, published by Freeman, 2003.
- Essential Discrete Mathematics for Computer Science, by T. Feil and J. Krone, published by Prentice Hall, 2003.
- Discrete Mathematics for Computing, by R. Haggerty, published by Addison Wesley, 2002.
- Discrete Structures, Logic, and Computability, 2nd edition, by J.L. Hein, published by Jones and Bartlett, 2002.
- Discrete Mathematics for Computer Scientists, by J. Truss, published by Addison Wesley, 1999.
- Discrete Mathematics with Applications, 3rd edition, by S. Epp, published by Brooks/Cole, 2004.
- Discrete Mathematics with Proof, by E. Gossett, published by Prentice Hall, 2003.
- Discrete Mathematics, 5th edition, by K.A. Ross and C.R.B. Wright, published by Prentice Hall, 2003.
- Discrete Mathematics, 4th edition, by J.A. Dossey, A.D. Otto, L.E. Spence, and C.V. Eynden, published by Addison Wesley, 2002.
- Mathematics: A Discrete Introduction, by E.R. Scheinerman, published by Brooks/Cole, 2000.
- Discrete Mathematics, by S. Washburn, T. Marlowe, and C.T. Ryan, published by Addison Wesley, 1999.
- Discrete Mathematics with Graph Theory, 2nd edition, by E.G. Goodaire and M.M. Parmenter, published by Prentice Hall, 2002.
- Discrete and Combinatorial Mathematics, 5th edition, by R.P. Grimaldi, published by Addison Wesley, 2004.
- Discrete Mathematics with Combinatorics, 2nd edition, by J.A. Anderson, published by Prentice Hall, 2004.
- Discrete Mathematics: Numbers and Beyond, by S. Barnett, published by Addison Wesley, 1998.

1.15 Policy on cheating

The instructor will put a great deal of effort into helping students to understand and to learn the material in the course. However, the instructor will not tolerate any form of cheating.

The following behaviour will be regarded as cheating (together with other acts that would normally be regarded as cheating in the broad sense of the term):

- Copying assignments
- Allowing another student to copy an assignment from you and present it as their own work
- Copying from another student during a test or exam
- Referring to notes, textbooks, etc. during a test or exam
- Talking during a test or an exam
- Not sitting at the pre-assigned seat during a test or exam
- Communicating with another student in any way during a test or exam
- Having access to the exam/test paper prior to the exam/test
- Asking a teaching assistant for the answer to a question during an exam/test
- Presenting another's work as your own
- Modifying answers after they have been marked
- Any other behaviour which attempts unfairly to give you an advantage over other students in the grade-assessment process
- Refusing to obey the instructions of the officer in charge of an examination.

Chapter 2

Discrete Structures Introduction[1]

2.1 Introduction to Discrete Structures

2.1.1 What is Discrete Mathematics?

Discrete mathematics is mathematics that deals with discrete objects. Discrete objects are those which are separated from (not connected to/distinct from) each other. Integers (aka whole numbers), rational numbers (ones that can be expressed as the quotient of two integers), automobiles, houses, people etc. are all discrete objects. On the other hand real numbers which include irrational as well as rational numbers are not discrete. As you know between any two different real numbers there is another real number different from either of them. So they are packed without any gaps and can not be separated from their immediate neighbors. In that sense they are not discrete. In this course we will be concerned with objects such as integers, propositions, sets, relations and functions, which are all discrete. We are going to learn concepts associated with them, their properties, and relationships among them among others.

2.1.2 Why Discrete Mathematics?

Let us first see why we want to be interested in the formal/theoretical approaches in computer science.

Some of the major reasons that we adopt formal approaches are 1) we can handle infinity or large quantity and indefiniteness with them, and 2) results from formal approaches are reusable. As an example, let us consider a simple problem of investment. Suppose that we invest $1,000 every year with expected return of 10% a year. How much are we going to have after 3 years, 5 years, or 10 years? The most naive way to find that out would be the brute force calculation. Let us see what happens to $1,000 invested at the beginning of each year for three years. First let us consider the $1,000 invested at the beginning of the first year. After one year it produces a return of $100. Thus at the beginning of the second year, $1,100, which is equal to $1,000 * (1 + 0.1), is invested. This $1,100 produces $110 at the end of the second year. Thus at the beginning of the third year we have $1,210, which is equal to $1,000 * (1 + 0.1)*(1 + 0.1), or $1,000 * (1 + 0.1)2. After the third year this gives us $1,000 * (1 + 0.1)3. Similarly we can see that the $1,000 invested at the beginning of the second year produces $1,000 * (1 + 0.1)2 at the end of the third year, and the $1,000 invested at the beginning of the third year becomes $1,000 * (1 + 0.1). Thus the total principal and return after three years is $1,000 * (1 + 0.1) + $1,000 * (1 + 0.1)2 + $1,000 * (1 + 0.1)3, which is equal to $3,641.

One can similarly calculate the principal and return for 5 years and for 10 years. It is, however, a long tedious calculation even with calculators. Further, what if you want to know the principal and return for some different returns than 10%, or different periods of time such as 15 years? You would have to do all these calculations all over again. We can avoid these tedious calculations considerably by noting the similarities in these problems and solving them in a more general way. Since all these problems ask for the result of

[1] This content is available online at <http://cnx.org/content/m14586/1.3/>.

investing a certain amount every year for certain number of years with a certain expected annual return, we use variables, say A, R and n, to represent the principal newly invested every year, the return ratio, and the number of years invested, respectively. With these symbols, the principal and return after n years, denoted by S, can be expressed as S = A(1 + R) + A(1 + R)2 + ... + A(1 + R)n. As well known, this S can be put into a more compact form by first computing S - (1 + R)S as

S = A ((1 + R)n + 1 - (1 + R)) / R .

Once we have it in this compact form, it is fairly easy to compute S for different values of A, R and n, though one still has to compute (1 + R)n + 1 . This simple formula represents infinitely many cases involving all different values of A, R and n. The derivation of this formula, however, involves another problem. When computing the compact form for S, S - (1 + R)S was computed using S = A(1 + R) + A(1 + R)2 + ... + A(1 + R)n . While this argument seems rigorous enough, in fact practically it is a good enough argument, when one wishes to be very rigorous, the ellipsis ... in the sum for S is not considered precise. You are expected to interpret it in a certain specific way. But it can be interpreted in a number of different ways. In fact it can mean anything. Thus if one wants to be rigorous, and absolutely sure about the correctness of the formula, one needs some other way of verifying it than using the ellipsis. Since one needs to verify it for infinitely many cases (infinitely many values of A, R and n), some kind of formal approach, abstracted away from actual numbers, is required.

Suppose now that somehow we have formally verified the formula successfully and we are absolutely sure that it is correct. It is a good idea to write a computer program to compute that S, especially with (1 + R)n + 1 to be computed. Suppose again that we have written a program to compute S. How can we know that the program is correct? As we know, there are infinitely many possible input values (that is, values of A, R and n). Obviously we can not test it for infinitely many cases. Thus we must take some formal approach. Related to the problem of correctness of computer programs, there is the well known "Halting Problem". This problem, if put into the context of program correctness, asks whether or not a given computer program stops on a given input after a finite amount of time. This problem is known to be unsolvable by computers. That is, no one can write a computer program to answer that question. It is known to be unsolvable. But, how can we tell it is unsolvable? How can we tell that such a program can not be written? You can not try all possible solution methods and see they all fail. You can not think of all (candidate) methods to solve the Halting Problem. Thus you need some kind of formal approaches here to avoid dealing with an extremely large number (if not infinite) of possibilities. Discrete mathematics is the foundation for the formal approaches. It discusses languages used in mathematical reasoning, basic concepts, and their properties and relationships among them. Though there is no time to cover them in this course, discrete mathematics is also concerned with techniques to solve certain types of problems such as how to count or enumerate quantities. The kind of counting problems includes: How many routes exist from point A to point B in a computer network? How much execution time is required to sort a list of integers in increasing order? What is the probability of winning a lottery? What is the shortest path from point A to point B in a computer network? ...

The subjects covered in this course include propositional logic, predicate logic, sets, relations, and functions, in particular growth of function. The first subject is logic. It is covered in Chapter 1 of the textbook. It is a language that captures the essence of our reasoning, and correct reasoning must follow the rules of this language. We start with logic of sentences called propositional logic, and study elements of logic, (logical) relationships between propositions, and reasoning. Then we learn a little more powerful logic called predicate logic. It allows us to reason with statements involving variables among others. In Chapter 1 we also study sets, relations between sets, and operations on sets. Just about everything is described based on sets, when rigor is required. It is the basis of every theory in computer science and mathematics. In Chapter 3 we learn mathematical reasoning, in particular recursive definitions and mathematical induction. There are sets, operations and functions that can be defined precisely by recursive definitions. Properties of those recursively defined objects can be established rigorously using proof by induction. Then in Chapter 6 we study relations. They are an abstraction of relations we are familiar with in everyday life such as husband-wife relation, parent-child relation and ownership relation. They are also one of the key concepts in the discussion of many subjects on computer and computation. For example, a database is viewed as

a set of relations and database query languages are constructed based on operations on relations and sets. Graphs are also covered briefly here. They are an example of discrete structures and they are one of the most useful models for computer scientists and engineers in solving problems. More in-depth coverage of graph can be found in Chapter 7. Finally back in Chapter 1 we study functions and their asymptotic behaviors. Functions are a special type of relation and basically the same kind of concept as the ones we see in calculus. However, function is one of the most important concepts in the discussion of many subjects on computer and computation such as data structures, database, formal languages and automata, and analysis of algorithms which is briefly covered in Chapter 2.

Chapter 3

Discrete Structures Problem Solving[1]

3.1 Problem Solving

3.1.1 Introduction

Everyone must have felt at least once in his or her life how wonderful it would be if we could solve a problem at hand preferably without much difficulty or even with some difficulties. Unfortunately the problem solving is an art at this point and there are no universal approaches one can take to solving problems. Basically one must explore possible avenues to a solution one by one until one comes across a right path to a solution. Thus generally speaking, there is guessing and hence an element of luck involved in problem solving. However, in general, as one gains experience in solving problems, one develops one's own techniques and strategies, though they are often intangible. Thus the guessing is not an arbitrary guessing but an educated one. In this chapter we are going to learn a framework for problem solving and get a glimpse of strategies that are often used by experts. They are based on the work of Polya. For further study, his book, and others such as Larson are recommended (but not required).

3.1.2 A Framework for Problem Solving

The following four phases can be identified in the process of solving problems: (1) Understanding the problem
(2) Making a plan of solution
(3) Carrying out the plan
(4) Looking back i.e. verifying

Each of the first two phases is going to be explained below a little more in detail. Phases (3) and (4) should be self-explanatory.

3.1.3 Understanding the Problem

Needless to say that if you do not understand the problem you can never solve it. It is also often true that if you really understand the problem, you can see a solution. Below are some of the things that can help us understand a problem. (1) Extract the principal parts of the problem.

The principal parts are:

For "find" type problems, such as "find the principal and return for a given investment", UNKNOWNS, DATA and CONDITIONS, and for "proof" type problems HYPOTHESIS and CONCLUSION. For examples that illustrate these, see examples below. Be careful about hidden assumptions, data and conditions.

(2) Consult definitions for unfamiliar (often even familiar) terminologies.
(3) Construct one or two simple example to illustrate what the problem says.

[1] This content is available online at <http://cnx.org/content/m15771/1.1/>.

3.1.4 Devising a Solution Plan

Where to start?

Start with the consideration of the principal parts: unknowns, data, and conditions for "find" problems, and hypothesis, and conclusion for "prove" problems.

What can I do?

Once you identify the principal parts and understand them, the next thing you can do is to consider the problem from various angles and seek contacts with previously acquired knowledge. The first thing you should do is to try to find facts that are related to the problem at hand. Relevant facts usually involve words that are the same as or similar to those in the given problem. It is also a good idea to try to recall previously solved similar problems.

What should I look for?

Look for a helpful idea that shows you the way to the end. Even an incomplete idea should be considered. Go along with it to a new situation, and repeat this process.

Some helpful tips

There is no single method that works for all problems. However, there is a wealth of heuristics we can try. Following are some of the often used heuristics. Add your own heuristics as you gain experience.

(1) Have I seen it before?

That is, do I know similar or related problems? Similar/related problems are ones with the same or a similar unknown or unknown may be different but the settings are the same or similar.

(2) Do a little analysis on relationships among data, conditions and unknowns, or between hypothesis and conclusion.

(3) What facts do I know related to the problem on hand?

These are facts on the subjects appearing in the problem. They often involve the same or similar words. It is very important that we know inference rules.

(4) Definitions: Make sure that you know the meaning of technical terms. This is obviously crucial to problem solving at any level. But especially at this level, if you know their meaning and understand the concepts, you can see a solution to most of the problems without much difficulty.

(5) Compose a wish list of intermediate goals and try to reach them.

(6) Have you used all the conditions/hypotheses? When you are looking for paths to a solution or trying to verify your solution, it is often a good idea to check whether or not you have used all the data/hypotheses. If you haven't, something is often amiss.

(7) Divide into cases: Sometimes if you divide your problem into a number of separate cases based on a property of objects appearing in the problem, it simplifies the problem and clear your mind. For example if the problem concerns integers, then you may want to divide it into two cases: one for even numbers and the other for odd numbers as. (8) Proof by contradiction: If you make an assumption, and that assumption produces a statement that does not make sense, then you must conclude that your assumption is wrong. For example, suppose that your car does not start. A number of things could be wrong. Let us assume for simplicity's sake that either the battery is dead or the starter is defective. So you first assume that the battery is dead and try to jump start it. If it doesn't start, you have a situation that does not make sense under your assumption of dead battery. That is, a good battery should start the car but it doesn't. So you conclude that your assumption is wrong. That is the battery is not the cause. Proof by contradiction follows that logic.

In this method we first assume that the assertion to be proven is not true and try to draw a contradiction i.e. something that is always false. If we produce a contradiction, then our assumption is wrong and therefore the assertion we are trying to prove is true.When you are stuck trying to justify some assertion, proof by contradiction is always a good thing to try.

(9) Transform/Restate the problem, then try (1) - (3) above.

(10) Working backward: In this approach, we start from what is required, such as conclusion or final (desired) form of an equation etc., and assume what is sought has been found. Then we ask from what antecedent the desired result could be derived. If the antecedent is found, then we ask from what antecedent

that antecedent could be obtained. ... We repeat this process until either the data/hypotheses are reached or some easy to solve problem is reached.

(11) Simplify the problem if possible. Take advantage of symmetries which often exist.

Keep in mind that your first try may not work. But don't get discouraged. If one approach doesn't work, try another. You have to keep trying different approaches, different ideas. As you gain experience, your problem solving skills improve and you tend to find the right approach sooner.

Let us now look at some examples to illustrate the topics discussed above.

3.1.5 Examples

Example 1

This is an example in which you can find a solution once you analyze and understand the unknowns and data.

Problem: A survey of TV viewers shows the following results:

To the question "Do you watch comedies?", 352 answered "Yes".,

To the question "Do you watch sports ?", 277 answered "Yes", and

To the question "Do you watch both comedies and sports ?", 129 answered "Yes".

Given these data, find, among people who watch at least one of comedies and sports, percentages of people who watch at least one of comedies and sports watch only comedies, only sports, and both comedies and sports.

Let us try to solve this problem following the framework presented above.

Understanding the Problem: This is a "find" type problem. So we try to identify unknowns, data and conditions.

The unknowns are the percentage of people who watch only comedies, the percentage of people who watch only sports, and the percentage of people who watch both comedies and sports.

The data are the three numbers: 352, 277 and 129, representing the number of people who watch comedies, sports, and both comedies and sports, respectively. Note that 352 includes people who watch both comedies and sports as well as people who watch only comedies. Similarly for 277.

The conditions are not explicitly given in the problem statement. But one can see that the percentages must add up to 100, and they must be nonnegative.

Devising a Solution Plan: Here we first examine the principal parts in detail.

First let us consider the unknowns in more detail. To calculate the percentage of the people who watch only comedies, for example, we need the number of people who watch at least one of comedies and sports, and the number of people who watch only comedies. Thus actually two unknowns are involved in each of the required percentages, and the real unknowns are the number of people in each of the categories, and the number of people who watch at least one of comedies and sports.

Next let us look at the data. First the number 352 is the number of people who watch comedies. But that is not necessarily that of the people who watch only comedies. It includes that and the number of people who watch both comedies and sports. Similarly for the second number 277.

Let us use symbols to represent each of the unknowns: Let C represent the number of people who watch only comedies, S that of the people who watch only sports, and T that of the people who watch at least one of those programs.

Then we have the following relationships among the unknowns:

$C + 129 = 352$

$S + 129 = 277$

$C + S + 129 = T$

From these equations we can easily obtain $C = 223$, $S = 148$, and $T = 500$. Thus the required percentages are 44.6%, 29.6%, and 25.8%, respectively.

All we had to do to solve this problem is to analyze relationships between the data and the unknowns, that is, nothing much beyond "understanding the problem".

Example 2

This is a problem which you can solve using similar known results.

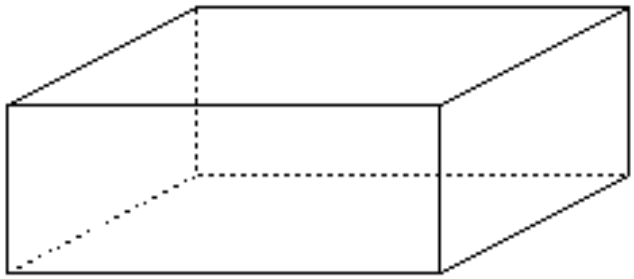
Problem: Find the (length of) diagonal of a rectangular parallelepiped given its length, width and height.

Again let us try to solve this problem following the framework presented above.

Understanding the Problem: This is a "find" type problem. So we try to identify unknowns, data and conditions.

The unknown is the diagonal of a rectangular parallelepiped, and the data are its length, width and height. Again there are no explicitly stated conditions. But the unknown and data must all be a positive number.

Before proceeding to the next phase, let us make sure that we understand the terminologies. First a rectangular parallelepiped is a box with rectangular faces like a cube except that the faces are not necessarily a square but a rectangle as shown in Figure 1.



Rectangular Parallelpiped

Figure 3.1

Next a diagonal of a rectangular parallelepiped is the line that connects its two vertices (corner points) that are not on the same plane. It is shown in Figure 2.

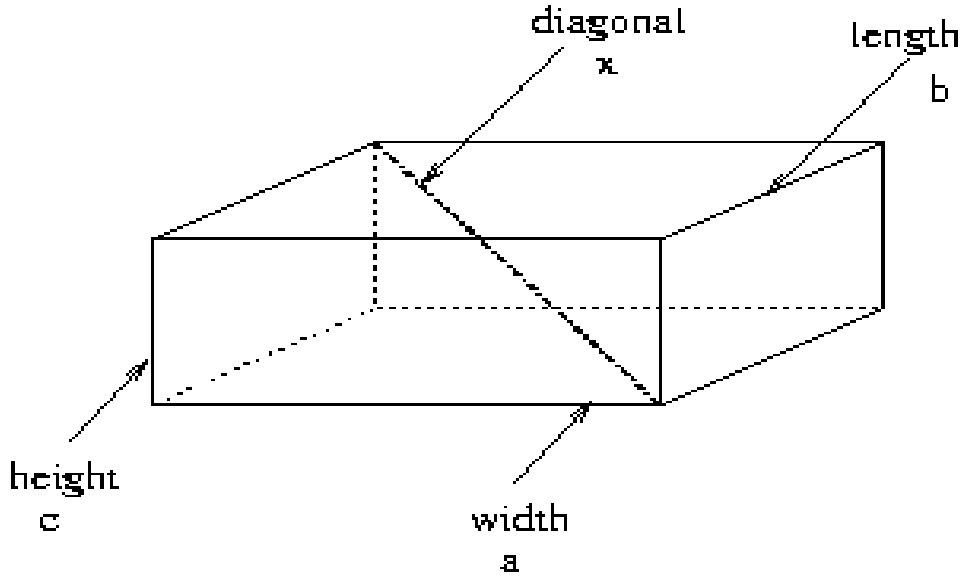


Figure 3.2

Devising a Solution Plan: Here we first try to find relevant facts. Relevant facts often involve the same or similar words or concepts. Since the unknown is a diagonal, we look for facts concerning diagonal. Note that drawing figures here is quite helpful. One of the facts that immediately comes to our mind in this problem is Pythagoras' theorem. It has to do with right triangles and is shown in Figure 3.

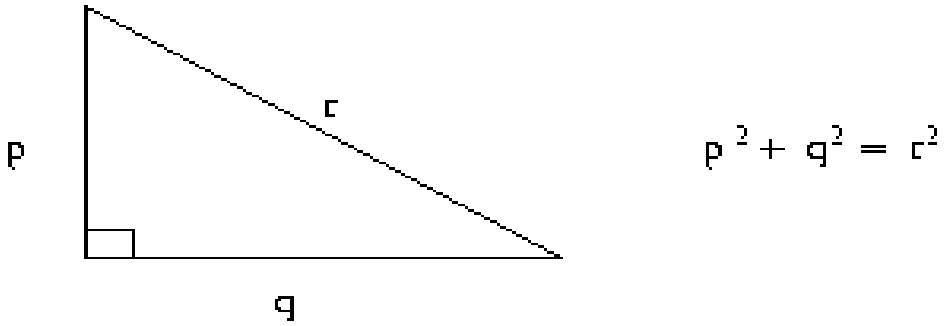


Figure 3.3

Let us try to see whether or not this theorem helps. To use this theorem, we need a right triangle involving a diagonal of a parallelepiped. As we can see in Figure 4, there is a right triangle with a diagonal x as its hypotenuse.

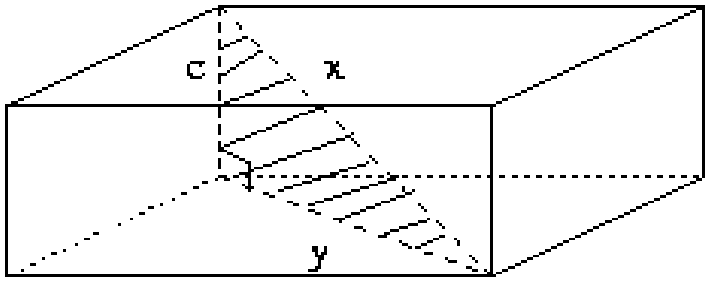


Figure 3.4

However, the triangle here involves two unknowns: x and y. Since x is what we are looking for, we need to find the value of y. To find y, we note another right triangle shown in Figure 5.

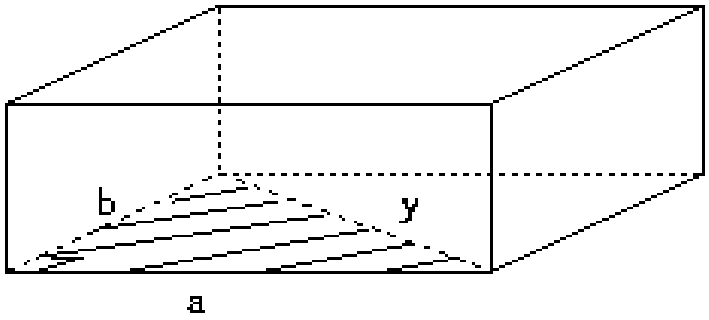


Figure 3.5

Applying Pythagoras' theorem again, we can obtain the value of y.
Thus $y2 = a2 + b2$
is obtained from the second triangle, and
$x2 = c2 + y2$
is derived from the first triangle.
From these two equations, we can find that x is equal to the positive square root of $a2 + b2 + c2$.
Example 3
This is a proof type problem and "proof by contradiction" is used.
Problem: Given that a, b, and c are odd integers, prove that equation $ax2 + bx + c = 0$ can not have a rational root.
Understanding the Problem: This is a "prove" type problem.
The hypothesis is that a, b, and c are odd integers, and the conclusion is that equation $ax2 + bx + c = 0$ can not have a rational root.
The hypothesis is straightforward. In the conclusion, "rational root" means a root, that is, the value of x that satisfies the equation, and that can be expressed as m/n, where m and n are integers. So the conclusion means that there is no number of the form m/n that satisfies the equation under the hypothesis.

Devising a Solution Plan: For this problem, let us try "proof by contradiction". When you are asked to prove the impossibility of an event or non-existence of certain things, this approach often is quite helpful.

Following the "proof by contradiction", let us assume that the conclusion is false, that is the equation ax2 + bx + c = 0 has a rational root m/n, where m and n are integers, when a, b, and c are odd integers. We can assume without loss of generality that m and n do not have any factors in common. Then

a(m/n)2 + b(m/n) + c = 0 . ——————— (1)

Let us try to derive a contradiction from this. First let us make this equation simpler, that is, let us get rid of fractions.

Since n is not equal to 0, multiplying the both sides of (1) by n2, we get

am2 + bmn + cn2 = 0. ——————— (2)

Since m is an integer, it is either even or odd. We are going to consider those cases one by one. That is "divide into cases".

Let us first consider the case when m is even. Then n is odd, since otherwise m and n have a common factor 2. Now am2 + bmn is even, and cn2 is odd. Hence am2 + bmn + cn2 can not be 0.

Next let us consider the case when m is odd. By an argument similar to the previous case, we can see that n is also odd. If m and n are odd, then am2, bmn, and cn2 are all odd, since a, b, and c are odd integers. However, the sum of three odd numbers can not be equal to 0.

Thus by assuming that the conclusion is false, we have arrived at a contradiction, that is m/n does not satisfy the equation. Hence our assumption must be wrong, and therefore the conclusion is correct.

Example 4

This is another proof type problem and "working backward" technique is used.

Problem: Prove that (a + b + c)2 $\leq$ 4(ab + bc + ca) , if a, b, c are three sides of a triangle. Understanding the Problem: This is a "prove" type problem.

The hypothesis is that a, b, and c are three sides of a triangle, and the conclusion is that inequality (a + b + c)2 $\leq$4(ab + bc + ca) holds.

Devising a Solution Plan: Here we try "Working Backward" heuristic. That is manipulate the conclusion possibly using the hypothesis and reduce it into something that is obviously true.

First by multiplying out the left hand side of the inequality, (a + b + c)2 = a2 + b2 + c2 + 2(ab + bc + ca).

Hence if a2 + b2 + c2 $\leq$ 2(ab + bc + ca) , then the conclusion holds.

Next, to see what we can try, note that we have not used the hypothesis yet, and see if it can help here.

It is well known that the sum of two sides of a triangle is greater than the third side. Hence a + b > c , b + c > a , and c + a > b hold.

From these we can obtain c(a + b) > c2 , a(b + c) > a2 , and b(c + a) > b2.

By adding these three inequalities, we get

a2 + b2 + c2 < a(b + c) + b(c + a) + c(a + b) = 2(ab + bc + ca).

Hence a2 + b2 + c2 < 2(ab + bc + ca).

Hence a2 + b2 + c2 $\leq$ 2(ab + bc + ca).

Hence (a + b + c)2 $\leq$4(ab + bc + ca) holds.

Example 5

This is a find type problem and "working backward" technique is used.

Problem: Given a 4 quart pail and a 9 quart pail, obtain 6 quarts of water in the 9 quart pail using these two pails. You can fill or empty the pails and you can have as much water as you want.

Understanding the Problem: This is a "find" type problem.

The problem is to obtain 6 quarts of water in the 9 quart pail using 4 quart and 9 quart pails as measures. You can fill either pail from the water source or from the other pail, and you can empty the pails any time.

Devising a Solution Plan: You can solve this in a number of different ways. Here we try "Working Backward" heuristic. It starts with the desired solution and work backward step by step. At each step we try to find a state that immediately precedes the current state such that we can reach the current state from that state with one simple operation such as filling a pail or emptying a pail in this problem. We repeat this

process until we reach some easily reachable state such as empty pails, full pails, one pail full and the other empty, etc.

Our solution to the original problem is obtained by traversing this process backward to the desired state.

Let us denote the 9 quart pail by A and the 4 quart pail by B for simplicity. In this problem, the desired state is to have 6 qts in A (Figure 6).

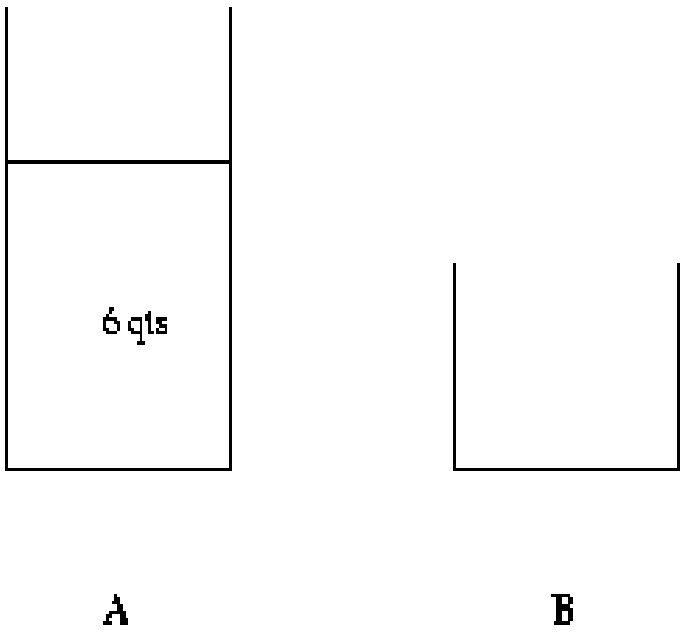


Figure 3.6

Thus in the first step of "working backward", we ask how we could get to the desired state with one operation.

As one can easily see if we could dump 3 qts from 9 qts in A, then we would have 6 qts in A. To be able to dump 3 qts from A we need 1 qt in B. Thus the state immediately preceding the current state is the one where A is full and B has 1 qt in it (Figure 7).

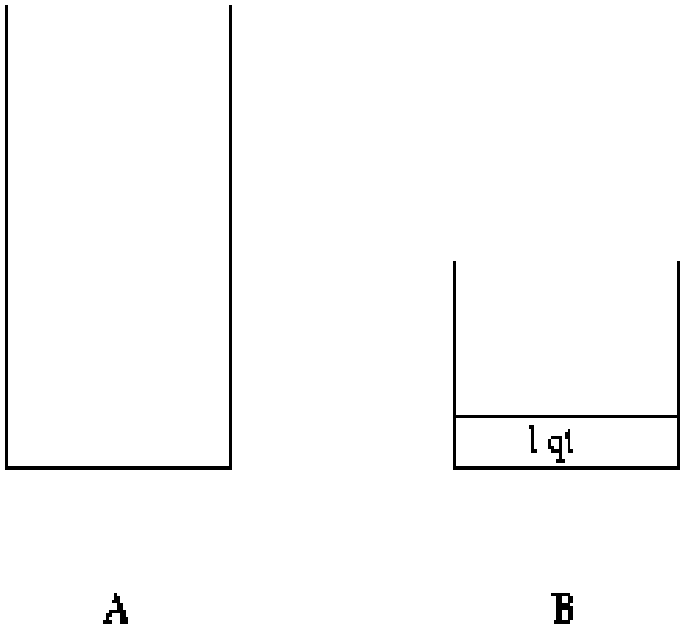


Figure 3.7

In the second step, the question we ask is how to get 1 qt in B. It does not look easy to get 1 qt in B. So let us see whether or not we can get 1 qt in A. If we have 1 qt in A, then we can certainly get 1 qt in B without any trouble. thus we might say that the third state is to have 1 qt in A (Figure 8).

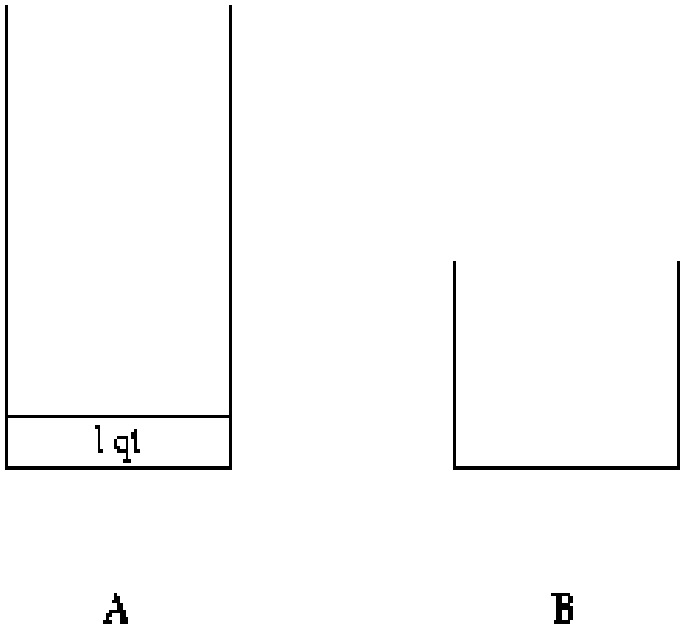


Figure 3.8

In the third step, the question we ask is how to get 1 qt in A. This is relatively easy to accomplish because all you have to do is to get rid of 8 qts from a full A, which can be done by emptying A twice into B.

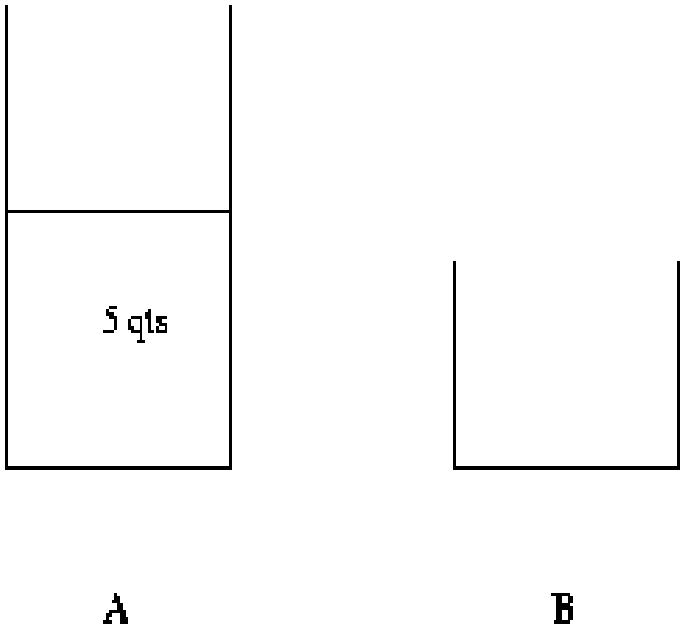


Figure 3.9

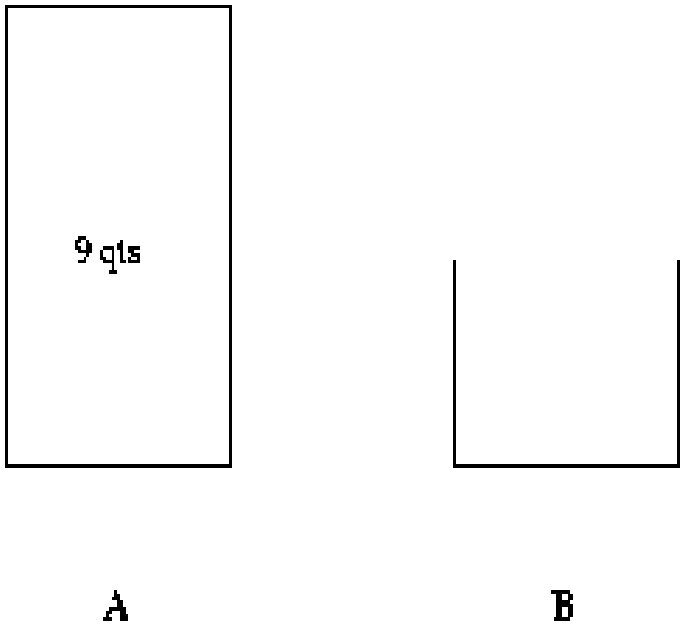


Figure 3.10

Since this state can be easily reached (all you have to do to get to this state is to fill A with water), we stop here. Our solution to the original problem is now obtained by going this process backward.

Thus first we fill up A. Then dump A into B leaving 5 qts in A (Figure 9). Then dump A into B again. This gives us 1 qt in A. Pour that into B. Then fill A and empty it into B. We now have 6 qts in A, which is what is required.

Example 6

Problem: A survey of TV viewers shows the following results:

To the question "Do you watch comedies?", 374 replied "Yes".

To the question "Do you watch sports?", 360 replied "Yes".

To the question "Do you watch detective stories?", 350 replied "Yes".

To the question "Do you watch comedies and sports?", 134 replied "Yes".

To the question "Do you watch comedies and detectives?", 96 replied "Yes".

To the question "Do you watch detectives and sports?", 241 replied "Yes".

To the question "Do you watch all three?", 37 replied "Yes".

Find the percentages of people who watch only comedies, only sports, only detective stories, comedies and sports, comedies and detectives, sports and detectives, and all three. By doing the same kind of analyses on the unknowns and data, one can find these percentages. The total number of people who watch at least one of these programs is

374 + 360 + 350 - 134 - 96 - 241 + 37 = 650,

because each of 134, 96 and 241 is counted twice into 374 + 360 + 350, and 37 is counted in three times and subtracted out three times in 374 + 360 + 350 - 134 - 96 - 241.

Similarly the number of those watching only comedies is

374 - 134 - 96 + 37 = 181,

the number of those watching only detectives is

350 - 241 - 96 + 37 = 50,
the number of those watching only sports is
360 - 134 - 241 + 37 = 22,
the number of those watching only comedies and detectives is
96 - 37 = 59,
the number of those watching only comedies and sports is
350 - 37 = 313,
the number of those watching only detectives and sports is
241 - 37 = 204.
The calculation of the percentages is omitted.

Example 7

300 people were surveyed on TV programs they watch and all 300 responded. 75 of them say they watch a regular sports program, 210 say they watch a regular comedy program, and 36 say they watch both. There is a special detective story program which conflicts with the regular sports and comedy programs. If those who watch regular programs do not want to miss them, how many people can watch the special?

To answer the question, all you have to do is to find the total number of people who watch at least one of the regular sports and comedy programs. Then 300 minus that number is the answer. Thus this problem is essentially the same as Example 1, and the answer is 300 - (75 + 210 - 36) = 51.

Example 8

300 people were surveyed. 50 of them watch sports, 140 watch comedies, and 134 do not watch either of them. Then how many of them watch both comedies and sports? The relationships among various groups of people in this problem are the same as those of Example 1 or 7. Only the data are slightly different.

Since 300 were surveyed and 134 do not watch either sports or comedies, 166 watch at least one. Hence 50 + 140 - 166 = 24 watch both. Thus 50 - 24 = 26 watch only sports, and 140 - 24 = 116 watch only comedies.

3.1.5.1 Reference

Polya: G. Polya, How to Solve It, **A New Aspect of Mathematical Method**, Second Ed., Princeton University Press, Princeton, NJ, 1985. Larson: L. C. Larson, Problem-Solving Through Problems, Springer-Verlag, New York, NY, 1983.

Chapter 4

Discrete Structures Logic[1]

4.1 Introduction to Logic

4.1.1 Logic

Logic is a language for reasoning. It is a collection of rules we use when doing logical reasoning. Human reasoning has been observed over centuries from at least the times of Greeks, and patterns appearing in reasoning have been extracted, abstracted, and streamlined. The foundation of the logic we are going to learn here was laid down by a British mathematician George Boole in the middle of the 19th century, and it was further developed and used in an attempt to derive all of mathematics by Gottlob Frege, a German mathematician, towards the end of the 19th century. A British philosopher/mathematician, Bertrand Russell, found a flaw in basic assumptions in Frege's attempt but he, together with Alfred Whitehead, developed Frege's work further and repaired the damage. The logic we study today is more or less along this line.

In logic we are interested in true or false of statements, and how the truth/falsehood of a statement can be determined from other statements. However, instead of dealing with individual specific statements, we are going to use symbols to represent arbitrary statements so that the results can be used in many similar but different cases. The formalization also promotes the clarity of thought and eliminates mistakes.

There are various types of logic such as logic of sentences (propositional logic), logic of objects (predicate logic), logic involving uncertainties, logic dealing with fuzziness, temporal logic etc. Here we are going to be concerned with propositional logic and predicate logic, which are fundamental to all types of logic.

4.1.2 Introduction to Propositional Logic

Propositional logic is a logic at the sentential level. The smallest unit we deal with in propositional logic is a sentence. We do not go inside individual sentences and analyze or discuss their meanings. We are going to be interested only in true or false of sentences, and major concern is whether or not the truth or falsehood of a certain sentence follows from those of a set of sentences, and if so, how. Thus sentences considered in this logic are not arbitrary sentences but are the ones that are true or false. This kind of sentences are called propositions.

4.1.2.1 Proposition

4.1.2.1.1 What Is Proposition?

Sentences considered in propositional logic are not arbitrary sentences but are the ones that are either true or false, but not both. This kind of sentences are called propositions. If a proposition is true, then we say it has a truth value of "true"; if a proposition is false, its truth value is "false".

[1] This content is available online at <http://cnx.org/content/m15773/1.1/>.

For example, "Grass is green", and "2 + 5 = 5" are propositions. The first proposition has the truth value of "true" and the second "false".

But "Close the door", and "Is it hot outside?" are not propositions. Also "x is greater than 2", where x is a variable representing a number, is not a proposition, because unless a specific value is given to x we can not say whether it is true or false, nor do we know what x represents.

Similarly "x = x" is not a proposition because we don't know what "x" represents hence what "=" means. For example, while we understand what "3 = 3" means, what does "Air is equal to air" or "Water is equal to water" mean? Does it mean a mass of air is equal to another mass or the concept of air is equal to the concept of air? We don't quite know what "x = x" mean. Thus we can not say whether it is true or not. Hence it is not a proposition.

4.1.2.1.2 Elements of Propositional Logic

Simple sentences which are true or false are basic propositions. Larger and more complex sentences are constructed from basic propositions by combining them with connectives. Thus propositions and connectives are the basic elements of propositional logic. Though there are many connectives, we are going to use the following five basic connectives here:

NOT, AND, OR, IF_THEN (or IMPLY), IF_AND_ONLY_IF.

They are also denoted by the symbols: $\neg$, $\bigwedge$,$\bigvee$,$\rightarrow$,$\leftrightarrow$,

respectively.

4.1.2.1.3 Truth Table

Often we want to discuss properties/relations common to all propositions. In such a case rather than stating them for each individual proposition we use variables representing an arbitrary proposition and state properties/relations in terms of those variables. Those variables are called a propositional variable. Propositional variables are also considered a proposition and called a proposition since they represent a proposition hence they behave the same way as propositions. A proposition in general contains a number of variables. For example (P $\bigvee$Q) contains variables P and Q each of which represents an arbitrary proposition. Thus a proposition takes different values depending on the values of the constituent variables. This relationship of the value of a proposition and those of its constituent variables can be represented by a table. It tabulates the value of a proposition for all possible values of its variables and it is called a truth table.

For example the following table shows the relationship between the values of P, Q and P$\bigvee$Q:

OR		
P	Q	(P $\bigvee$Q)
F	F	F
F	T	T
T	F	T
T	T	T

Table 4.1

In the table, F represents truth value false and T true. This table shows that P$\bigvee$Q is false if P and Q are both false, and it is true in all the other cases.

4.1.2.1.4 Meaning of the Connectives

Let us define the meaning of the five connectives by showing the relationship between the truth value (i.e. true or false) of composite propositions and those of their component propositions. They are going to be

shown using truth table. In the tables P and Q represent arbitrary propositions, and true and false are represented by T and F, respectively.

NOT	
P	¬P
T	F
F	T

Table 4.2

This table shows that if P is true, then (¬P) is false, and that if P is false, then (¬P) is true.

AND		
P	Q	(P ⋀Q)
F	F	F
F	T	F
T	F	F
T	T	T

Table 4.3

This table shows that (P⋀Q) is true if both P and Q are true, and that it is false in any other case. Similarly for the rest of the tables.

OR		
P	Q	(P ⋁Q)
F	F	F
F	T	T
T	F	T
T	T	T

Table 4.4

IMPLIES		
P	Q	(P→Q)
F	F	T
F	T	T
T	F	F
T	T	T

Table 4.5

When P→Q is always true, we express that by P ⇒Q. That is P ⇒Q is used when proposition P always implies proposition Q regardless of the value of the variables in them.

IF AND ONLY IF		
P	Q	(P ↔Q)
F	F	T
F	T	F
T	F	F
T	T	T

Table 4.6

When P ↔Q is always true, we express that by P ⇔Q. That is ⇔is used when two propositions always take the same value regardless of the value of the variables in them.

4.1.2.1.5 Construction of Complex Propositions

First it is informally shown how complex propositions are constructed from simple ones. Then more general way of constructing propositions is given.

In everyday life we often combine propositions to form more complex propositions without paying much attention to them. For example combining "Grass is green", and "The sun is red" we say something like "Grass is green and the sun is red", "If the sun is red, grass is green", "The sun is red and the grass is not green" etc. Here "Grass is green", and "The sun is red" are propositions, and form them using connectives "and", "if... then ..." and "not" a little more complex propositions are formed. These new propositions can in turn be combined with other propositions to construct more complex propositions. They then can be combined to form even more complex propositions. This process of obtaining more and more complex propositions can be described more generally as follows:

Let X and Y represent arbitrary propositions. Then [¬X], [X⋀Y], [X⋁Y], [X→Y], and [X↔Y] are propositions.

Note that X and Y here represent an arbitrary proposition. This is actually a part of more rigorous definition of proposition which we see later.

Example : [P → [Q ⋁ R]] is a proposition and it is obtained by first constructing [Q ⋁ R] by applying [X ⋁ Y] to propositions Q and R considering them as X and Y, respectively, then by applying [X→Y] to the two propositions P and [Q ⋁ R] considering them as X and Y, respectively.

Note: Rigorously speaking X and Y above are place holders for propositions, and so they are not exactly a proposition. They are called a propositional variable, and propositions formed from them using connectives are called a propositional form. However, we are not going to distinguish them here, and both specific propositions such as "2 is greater than 1" and propositional forms such as (P ⋁Q) are going to be called a proposition.

4.1.2.1.6 Converse and Contrapositive

For the proposition P→Q, the proposition Q→P is called its converse, and the proposition ¬ Q→ ¬ P is called its contrapositive.

For example for the proposition "If it rains, then I get wet",

Converse: If I get wet, then it rains.

Contrapositive: If I don't get wet, then it does not rain.

The converse of a proposition is not necessarily logically equivalent to it, that is they may or may not take the same truth value at the same time.

On the other hand, the contrapositive of a proposition is always logically equivalent to the proposition. That is, they take the same truth value regardless of the values of their constituent variables. Therefore, "If it rains, then I get wet." and "If I don't get wet, then it does not rain." are logically equivalent. If one is true then the other is also true, and vice versa.

4.1.2.2 From English to Proposition

4.1.2.2.1 If_Then Variations

If-then statements appear in various forms in practice. The following list presents some of the variations. These are all logically equivalent, that is as far as true or false of statement is concerned there is no difference between them. Thus if one is true then all the others are also true, and if one is false all the others are false.

- If p, then q.
- p implies q.
- If p, q.
- p only if q.
- p is sufficient for q.
- q if p.
- q whenever p.
- q is necessary for p.
- It is necessary for p that q.

For instance, instead of saying "If she smiles then she is happy", we can say "If she smiles, she is happy", "She is happy whenever she smiles", "She smiles only if she is happy" etc. without changing their truth values.

"Only if" can be translated as "then". For example, "She smiles only if she is happy" is equivalent to "If she smiles, then she is happy".

Note that "She smiles only if she is happy" means "If she is not happy, she does not smile", which is the contrapositive of "If she smiles, she is happy". You can also look at it this way: "She smiles only if she is happy" means "She smiles only when she is happy". So any time you see her smile you know she is happy. Hence "If she smiles, then she is happy". Thus they are logically equivalent.

Also "If she smiles, she is happy" is equivalent to "It is necessary for her to smile that she is happy". For "If she smiles, she is happy" means "If she smiles, she is always happy". That is, she never fails to be happy when she smiles. "Being happy" is inevitable consequence/necessity of "smile". Thus if "being happy" is missing, then "smile" can not be there either. "Being happy" is necessary "for her to smile" or equivalently "It is necessary for her to smile that she is happy".

4.1.2.2.2 From English to Proposition

As we are going to see in the next section, reasoning is done on propositions using inference rules. For example, if the two propositions "if it snows, then the school is closed", and "it snows" are true, then we can conclude that "the school is closed" is true. In everyday life, that is how we reason.

To check the correctness of reasoning, we must check whether or not rules of inference have been followed to draw the conclusion from the premises. However, for reasoning in English or in general for reasoning in a natural language, that is not necessarily straightforward and it often encounters some difficulties. Firstly, connectives are not necessarily easily identified as we can get a flavor of that from the previous topic on variations of if_then statements. Secondly, if the argument becomes complicated involving many statements in a number of different forms twisted and tangled up, it can easily get out of hand unless it is simplified in some way.

One solution for that is to use symbols (and mechanize it). Each sentence is represented by symbols representing building block sentences, and connectives. For example, if P represents "it snows" and Q represents "the school is closed", then the previous argument can be expressed as

[[P → Q] ⋀ P] → Q,
or
P → Q
P

Q This representation is concise, much simpler and much easier to deal with. In addition today there are a number of automatic reasoning systems and we can verify our arguments in symbolic form using them. One such system called TPS is used for reasoning exercises in this course. For example, we can check the correctness of our argument using it.

To convert English statements into a symbolic form, we restate the given statements using the building block sentences, those for which symbols are given, and the connectives of propositional logic (not, and, or, if_then, if_and_only_if), and then substitute the symbols for the building blocks and the connectives. For example, let P be the proposition "It is snowing", Q be the proposition "I will go the beach", and R be the proposition "I have time".

Then first "I will go to the beach if it is not snowing" is restated as "If it is not snowing, I will go to the beach". Then symbols P and Q are substituted for the respective sentences to obtain ∼P → Q.

Similarly, "It is not snowing and I have time only if I will go to the beach" is restated as "If it is not snowing and I have time, then I will go to the beach", and it is translated as (∼P ⋀ R) → Q.

4.1.2.3 Reasoning with Propositions

4.1.2.3.1 Introduction to Reasoning

Logical reasoning is the process of drawing conclusions from premises using rules of inference. Here we are going to study reasoning with propositions. Later we are going to see reasoning with predicate logic, which allows us to reason about individual objects. However, inference rules of propositional logic are also applicable to predicate logic and reasoning with propositions is fundamental to reasoning with predicate logic.

These inference rules are results of observations of human reasoning over centuries. Though there is nothing absolute about them, they have contributed significantly in the scientific and engineering progress the mankind have made. Today they are universally accepted as the rules of logical reasoning and they should be followed in our reasoning.

Since inference rules are based on identities and implications, we are going to study them first. We start with three types of proposition which are used to define the meaning of "identity" and "implication".

Some propositions are always true regardless of the truth value of its component propositions. For example (P ⋁¬P) is always true regardless of the value of the proposition P.

A proposition that is always true called a tautology.

There are also propositions that are always false such as (P ⋀¬P). Such a proposition is called a contradiction.

A proposition that is neither a tautology nor a contradiction is called a contingency. For example (P ⋁Q) is a contingency.

These types of propositions play a crucial role in reasoning. In particular every inference rule is a tautology as we see in identities and implications.

4.1.2.4 Identities

From the definitions (meaning) of connectives, a number of relations between propositions which are useful in reasoning can be derived. Below some of the often encountered pairs of logically equivalent propositions, also called identities, are listed.

These identities are used in logical reasoning. In fact we use them in our daily life, often more than one at a time, without realizing it.

If two propositions are logically equivalent, one can be substituted for the other in any proposition in which they occur without changing the logical value of the proposition.

Below $\Leftrightarrow$ corresponds to $\leftrightarrow$ and it means that the equivalence is always true (a tautology), while $\leftrightarrow$ means the equivalence may be false in some cases, that is in general a contingency.

That these equivalences hold can be verified by constructing truth tables for them.

First the identities are listed, then examples are given to illustrate them.

List of Identities:

1. $P \Leftrightarrow (P \bigvee P)$ —— idempotence of $\bigvee$
2. $P \Leftrightarrow (P \bigwedge P)$ —— idempotence of $\bigwedge$
3. $(P \bigvee Q) \Leftrightarrow (Q \bigvee P)$ —— commutativity of $\bigvee$
4. $(P \bigwedge Q) \Leftrightarrow (Q \bigwedge P)$ —— commutativity of $\bigwedge$
5. $[(P \bigvee Q) \bigvee R] \Leftrightarrow [P \bigvee (Q \bigvee R)]$ —— associativity of $\bigvee$
6. $[(P \bigwedge Q) \bigwedge R] \Leftrightarrow [P \bigwedge (Q \bigwedge R)]$ —— associativity of $\bigwedge$
7. $\neg(P \bigvee Q) \Leftrightarrow (\neg P \bigwedge \neg Q)$ —— DeMorgan's Law
8. $\neg(P \bigwedge Q) \Leftrightarrow (\neg P \bigvee \neg Q)$ —— DeMorgan's Law
9. $[P \bigwedge (Q \bigvee R] \Leftrightarrow [(P \bigwedge Q) \bigvee (P \bigwedge R)]$ —— distributivity of $\bigwedge$over $\bigvee$
10. $[P \bigvee (Q \bigwedge R] \Leftrightarrow [(P \bigvee Q) \bigwedge (P \bigvee R)]$ —— distributivity of $\bigvee$over $\bigwedge$
11. $(P \bigvee \text{True}) \Leftrightarrow \text{True}$
12. $(P \bigwedge \text{False}) \Leftrightarrow \text{False}$
13. $(P \bigvee \text{False}) \Leftrightarrow P$
14. $(P \bigwedge \text{True}) \Leftrightarrow P$
15. $(P \bigvee \neg P) \Leftrightarrow \text{True}$
16. $(P \bigwedge \neg P) \Leftrightarrow \text{False}$
17. $P \Leftrightarrow \neg(\neg P)$ —— double negation
18. $(P \rightarrow Q) \Leftrightarrow (\neg P \bigvee Q)$ —— implication
19. $(P \leftrightarrow Q) \Leftrightarrow [(P \rightarrow Q) \bigwedge (Q \rightarrow P)]$—— equivalence
20. $[(P \bigwedge Q) \rightarrow R] \Leftrightarrow [P \rightarrow (Q \rightarrow R)]$ —— exportation
21. $[(P \rightarrow Q) \bigwedge (P \rightarrow \neg Q)] \Leftrightarrow \neg P$ —— absurdity
22. $(P \rightarrow Q) \Leftrightarrow (\neg Q \rightarrow \neg P)$ —— contrapositive

Let us see some example statements in English that illustrate these identities.

Examples: 1. $P \Leftrightarrow (P \bigvee P)$ —— idempotence of $\bigvee$

What this says is, for example, that "Tom is happy." is equivalent to "Tom is happy or Tom is happy". This and the next identity are rarely used, if ever, in everyday life. However, these are useful when manipulating propositions in reasoning in symbolic form.

2. $P \Leftrightarrow (P \bigwedge P)$ —— idempotence of $\bigwedge$

Similar to 1. above.

3. $(P \bigvee Q) \Leftrightarrow (Q \bigvee P)$ —— commutativity of $\bigvee$

What this says is, for example, that "Tom is rich or (Tom is) famous." is equivalent to "Tom is famous or (Tom is) rich".

4. $(P \bigwedge Q) \Leftrightarrow (Q \bigwedge P)$ —— commutativity of $\bigwedge$

What this says is, for example, that "Tom is rich and (Tom is) famous." is equivalent to "Tom is famous and (Tom is) rich".

5. $[(P \bigvee Q) \bigvee R] \Leftrightarrow [P \bigvee (Q \bigvee R)]$ —— associativity of $\bigvee$

What this says is, for example, that "Tom is rich or (Tom is) famous, or he is also happy." is equivalent to "Tom is rich, or he is also famous or (he is) happy".

6. $[(P \bigwedge Q) \bigwedge R] \Leftrightarrow [P \bigwedge (Q \bigwedge R)]$ —— associativity of $\bigwedge$

Similar to 5. above.

7. $\neg(P \bigvee Q) \Leftrightarrow (\neg P \bigwedge \neg Q)$ —— DeMorgan's Law

For example, "It is not the case that Tom is rich or famous." is true if and only if "Tom is not rich and he is not famous."

8. $\neg(P \bigwedge Q) \Leftrightarrow (\neg P \bigvee \neg Q)$ —— DeMorgan's Law

For example, "It is not the case that Tom is rich and famous." is true if and only if "Tom is not rich or he is not famous."

9. [P $\bigwedge$(Q $\bigvee$R] $\Leftrightarrow$[(P $\bigwedge$Q) $\bigvee$(P $\bigwedge$R)] —— distributivity of $\bigwedge$ over $\bigvee$

What this says is, for example, that "Tom is rich, and he is famous or (he is) happy." is equivalent to "Tom is rich and (he is) famous, or Tom is rich and (he is) happy".

10. [P $\bigvee$(Q $\bigwedge$R] $\Leftrightarrow$[(P $\bigvee$Q) $\bigwedge$(P $\bigvee$R)] —— distributivity of $\bigvee$over $\bigwedge$

Similarly to 9. above, what this says is, for example, that "Tom is rich, or he is famous and (he is) happy." is equivalent to "Tom is rich or (he is) famous, and Tom is rich or (he is) happy".

11. (P $\bigvee$True) $\Leftrightarrow$ True. Here True is a proposition that is always true. Thus the proposition (P $\bigvee$True) is always true regardless of what P is.

This and the next three identities, like identities 1 and 2, are rarely used, if ever, in everyday life. However, these are useful when manipulating propositions in reasoning in symbolic form.

12. (P $\bigwedge$False) $\Leftrightarrow$False

13. (P $\bigvee$False) $\Leftrightarrow$P

14. (P $\bigwedge$True) $\Leftrightarrow$P

15. (P $\bigvee\neg$P) $\Leftrightarrow$True

What this says is that a statement such as "Tom is 6 foot tall or he is not 6 foot tall." is always true.

16. (P $\bigwedge\neg$P) $\Leftrightarrow$False

What this says is that a statement such as "Tom is 6 foot tall and he is not 6 foot tall." is always false.

17. P $\Leftrightarrow\neg(\neg$ P) —— double negation

What this says is, for example, that "It is not the case that Tom is not 6 foot tall." is equivalent to "Tom is 6 foot tall."

18. (P $\rightarrow$Q) $\Leftrightarrow(\neg$ P $\bigvee$Q) —— implication

For example, the statement "If I win the lottery, I will give you a million dollars." is not true, that is, I am lying, if I win the lottery and don't give you a million dollars. It is true in all the other cases. Similarly, the statement "I don't win the lottery or I give you a million dollars." is false, if I win the lottery and don't give you a million dollars. It is true in all the other cases. Thus these two statements are logically equivalent.

19. (P $\leftrightarrow$Q) $\Leftrightarrow$[(P $\rightarrow$Q) $\bigwedge$(Q $\rightarrow$P)]—— equivalence

What this says is, for example, that "Tom is happy if and only if he is healthy." is logically equivalent to ""if Tom is happy then he is healthy, and if Tom is healthy he is happy."

20. [(P $\bigwedge$Q) $\rightarrow$R] $\Leftrightarrow$[P $\rightarrow$(Q $\rightarrow$R)] —— exportation

For example, "If Tom is healthy, then if he is rich, then he is happy." is logically equivalent to "If Tom is healthy and rich, then he is happy."

21. [(P $\rightarrow$Q) $\bigwedge$(P$\rightarrow\neg$Q)] $\Leftrightarrow\neg$P —— absurdity

For example, if "If Tom is guilty then he must have been in that room." and "If Tom is guilty then he could not have been in that room." are both true, then there must be something wrong about the assumption that Tom is guilty.

22. (P $\rightarrow$Q) $\Leftrightarrow(\neg$Q $\rightarrow\neg$P) —— contrapositive

For example, "If Tom is healthy, then he is happy." is logically equivalent to "If Tom is not happy, he is not healthy."

The identities 1 $\sim$ 16 listed above can be paired by duality relation, which is defined below, as 1 and 2, 3 and 4, ..., 15 and 16. That is 1 and 2 are dual to each other, 3 and 4 are dual to each other, ... Thus if you know one of a pair, you can obtain the other of the pair by using the duality.

4.1.2.4.1 Dual of Proposition

Let X be a proposition involving only $\neg$, $\bigwedge$, and $\bigvee$ as a connective. Let X* be the proposition obtained from X by replacing $\bigwedge$ with $\bigvee$, $\bigvee$with $\bigwedge$, T with F, and F with T. Then X* is called the dual of X.

For example, the dual of [P $\bigwedge$Q] $\bigvee$P is [P $\bigvee$Q] $\bigwedge$P, and the dual of [$\neg$ P $\bigwedge$Q] $\bigvee\neg$[T $\bigwedge\neg$R] is [$\neg$ P $\bigvee$Q] $\bigwedge\neg$[F $\bigvee\neg$R] .

Property of Dual: If two propositions P and Q involving only $\neg$, $\bigwedge$, and $\bigvee$ as connectives are equivalent, then their duals P* and Q* are also equivalent.

4.1.2.4.2 Examples of Use of Identities

Here a few examples are presented to show how the identities in section Identities can be used to prove some useful results.

1. $\neg(P \rightarrow Q) \Leftrightarrow(P \bigwedge \neg Q)$

What this means is that the negation of "if P then Q" is "P but not Q". For example, if you said to someone "If I win a lottery, I will give you $100,000." and later that person says "You lied to me." Then what that person means is that you won the lottery but you did not give that person $100,000 you promised.

To prove this, first let us get rid of $\rightarrow$ using one of the identities: $(P \rightarrow Q) \Leftrightarrow(\neg P \bigvee Q)$.

That is, $\neg(P \rightarrow Q) \Leftrightarrow \neg(\neg P \bigvee Q)$.

Then by De Morgan, it is equivalent to $\neg\neg P \bigwedge \neg Q$, which is equivalent to $P \bigwedge \neg Q$, since the double negation of a proposition is equivalent to the original proposition as seen in the identities.

2. $P \bigvee(P \bigwedge Q) \Leftrightarrow P$ — Absorption

What this tells us is that $P \bigvee(P \bigwedge Q)$ can be simplified to P, or if necessary P can be expanded into $P \bigvee(P \bigwedge Q)$.

To prove this, first note that $P \Leftrightarrow(P \bigwedge T)$.

Hence

$P \bigvee(P \bigwedge Q)$

$\Leftrightarrow(P \bigwedge T) \bigvee(P \bigwedge Q)$

$\Leftrightarrow P \bigwedge(T \bigvee Q)$, by the distributive law.

$\Leftrightarrow(P \bigwedge T)$, since $(T \bigvee Q) \Leftrightarrow T$.

$\Leftrightarrow P$, since $(P \bigwedge T) \Leftrightarrow P$.

Note that by the duality

$P \bigwedge(P \bigvee Q) \Leftrightarrow P$ also holds.

4.1.2.4.3 Implications

The following implications are some of the relationships between propositions that can be derived from the definitions (meaning) of connectives. $\Rightarrow$ below corresponds to $\rightarrow$ and it means that the implication always holds. That is it is a tautology.

These implications are used in logical reasoning. When the right hand side of these implications is substituted for the left hand side appearing in a proposition, the resulting proposition is implied by the original proposition, that is, one can deduce the new proposition from the original one.

First the implications are listed, then examples to illustrate them are given. List of Implications:

1. $P \Rightarrow (P \bigvee Q)$ — addition
2. $(P \bigwedge Q) \Rightarrow P$ — simplification
3. $[P \bigwedge (P \rightarrow Q] \Rightarrow Q$ — modus ponens
4. $[(P \rightarrow Q) \bigwedge \neg Q] \Rightarrow \neg P$ — modus tollens
5. $[\neg P \bigwedge (P \bigvee Q] \Rightarrow Q$ — disjunctive syllogism
6. $[(P \rightarrow Q) \bigwedge (Q \rightarrow R)] \Rightarrow (P \rightarrow R)$ — hypothetical syllogism
7. $(P \rightarrow Q) \Rightarrow [(Q \rightarrow R) \rightarrow (P \rightarrow R)]$
8. $[(P \rightarrow Q) \bigwedge (R \rightarrow S)] \Rightarrow [(P \bigwedge R) \rightarrow (Q \bigwedge S)]$
9. $[(P \leftrightarrow Q) \bigwedge (Q \leftrightarrow R)] \Rightarrow (P \leftrightarrow R)$

Examples:

1. $P \Rightarrow (P \bigvee Q)$ — addition

For example, if the sun is shining, then certainly the sun is shining or it is snowing. Thus

"if the sun is shining, then the sun is shining or it is snowing." "If $0 < 1$, then $0 \leq 1$ or a similar statement is also often seen.

2. $(P \bigwedge Q) \Rightarrow P$ — simplification

For example, if it is freezing and (it is) snowing, then certainly it is freezing. Thus "If it is freezing and (it is) snowing, then it is freezing."

3. $[P \bigwedge (P \rightarrow Q] \Rightarrow Q$ — modus ponens

For example, if the statement "If it snows, the schools are closed" is true and it actually snows, then the schools are closed.

This implication is the basis of all reasoning. Theoretically, this is all that is necessary for reasoning. But reasoning using only this becomes very tedious.

4. $[(P \rightarrow Q) \bigwedge \neg Q] \Rightarrow \neg P$ —— modus tollens

For example, if the statement "If it snows, the schools are closed" is true and the schools are not closed, then one can conclude that it is not snowing. Note that this can also be looked at as the application of the contrapositive and modus ponens. That is, $(P \rightarrow Q)$ is equivalent to $(\neg Q) \rightarrow (\neg P)$. Thus if in addition $\neg Q$ holds, then by the modus ponens, $\neg P$ is concluded.

5. $[\neg P \bigwedge (P \bigvee Q] \Rightarrow Q$ —— disjunctive syllogism

For example, if the statement "It snows or (it) rains." is true and it does not snow, then one can conclude that it rains.

6. $[(P \rightarrow Q) \bigwedge (Q \rightarrow R)] \Rightarrow (P \rightarrow R)$ —— hypothetical syllogism

For example, if the statements "If the streets are slippery, the school buses can not be operated." and "If the school buses can not be operated, the schools are closed." are true, then the statement "If the streets are slippery, the schools are closed." is also true.

7. $(P \rightarrow Q) \Rightarrow [(Q \rightarrow R) \rightarrow (P \rightarrow R)]$

This is actually the hypothetical syllogism in another form. For by considering $(P \rightarrow Q)$ as a proposition S, $(Q \rightarrow R)$ as a proposition T, and $(P \rightarrow R)$ as a proposition U in the hypothetical syllogism above, and then by applying the "exportation" from the identities, this is obtained.

8. $[(P \rightarrow Q) \bigwedge (R \rightarrow S)] \Rightarrow [(P \bigwedge R) \rightarrow (Q \bigwedge S)]$

For example, if the statements "If the wind blows hard, the beach erodes." and "If it rains heavily, the streets get flooded." are true, then the statement "If the wind blows hard and it rains heavily, then the beach erodes and the streets get flooded." is also true.

9. $[(P \leftrightarrow Q) \bigwedge (Q \leftrightarrow R)] \Rightarrow (P \leftrightarrow R)$

This just says that the logical equivalence is transitive, that is, if P and Q are equivalent, and if Q and R are also equivalent, then P and R are equivalent.

4.1.2.4.4 Reasoning with Propositions

Logical reasoning is the process of drawing conclusions from premises using rules of inference. The basic inference rule is modus ponens. It states that if both $P \rightarrow Q$ and P hold, then Q can be concluded, and it is written as

P

$P \rightarrow Q$

Q

Here the lines above the dotted line are premises and the line below it is the conclusion drawn from the premises.

For example if "if it rains, then the game is not played" and "it rains" are both true, then we can conclude that the game is not played.

In addition to modus ponens, one can also reason by using identities and implications.

If the left (right) hand side of an identity appearing in a proposition is replaced by the right(left) hand side of the identity, then the resulting proposition is logically equivalent to the original proposition. Thus the new proposition is deduced from the original proposition. For example in the proposition $P \bigwedge (Q \rightarrow R)$, $(Q \rightarrow R)$ can be replaced with $(\neg Q \bigvee R)$ to conclude $P \bigwedge (\neg Q \bigvee R)$, since $(Q \rightarrow R) \Leftrightarrow (\neg Q \bigvee R)$

Similarly if the left (right) hand side of an implication appearing in a proposition is replaced by the right(left) hand side of the implication, then the resulting proposition is logically implied by the original proposition. Thus the new proposition is deduced from the original proposition.

The tautologies listed as "implications" can also be considered inference rules as shown below.

Rules of Inference	Tautological Form	Name
P ——————P ⋁Q	P ⇒(P ⋁Q)	addition
P ⋀Q ——————P	(P ⋀Q) ⇒P	simplification
P P→Q ——————Q	[P ⋀(P→Q)] ⇒Q	modus ponens
¬Q P→Q ——————¬P	[¬Q ⋀(P→Q)] ⇒¬P	modus tollens
P ⋁Q ¬P ——————Q	[(P ⋁Q) ⋀¬P] ⇒Q	disjunctive syllogism
P →Q Q→R ——————P →R	[(P →Q) ⋀(Q →R)] ⇒[P→R]	hypothetical syllogism
P Q ——————P ⋀Q		conjunction
(P→Q) ⋀(R→S) P ⋁R ——————Q ⋁S	[(P→Q) ⋀(R→S) ⋀(P ⋁R)] ⇒[Q ⋁S]	constructive dilemma
(P→Q) ⋀(R→S) ¬Q ⋁¬S ——————¬P ⋁¬R	[(P→Q) ⋀(R→S) ⋀(¬Q ⋁¬S)] ⇒[¬P ⋁¬R]	destructive dilemma

Table 4.7

4.1.2.4.5 Example of Inferencing

Consider the following argument:

1. Today is Tuesday or Wednesday.

2. But it can't be Wednesday, since the doctor's office is open today, and that office is always closed on Wednesdays.

3. Therefore today must be Tuesday.

This sequence of reasoning (inferencing) can be represented as a series of application of modus ponens to the corresponding propositions as follows.

The modus ponens is an inference rule which deduces Q from P → Q and P.

T: Today is Tuesday.

W: Today is Wednesday.

D: The doctor's office is open today.

C: The doctor's office is always closed on Wednesdays.

The above reasoning can be represented by propositions as follows.

1. T ⋁ W

2. D

C

———

∼W

———

3. T

To see if this conclusion T is correct, let us first find the relationship among C, D, and W:

C can be expressed using D and W. That is, restate C first as the doctor's office is always closed if it is Wednesday. Then C ↔ (W → ∼D) Thus substituting (W → ∼D) for C, we can proceed as follows.

D

W → ∼D

———

∼W

which is correct by modus tollens.

From this ∼W combined with T V W of 1. above,

∼W

T $\bigvee$ W

T

which is correct by disjunctive syllogism.

Thus we can conclude that the given argument is correct.

To save space we also write this process as follows eliminating one of the ∼W's:

D

W $\rightarrow$ ∼D

∼W

T $\bigvee$ W

T

4.1.2.4.6 Proof of Identities

All the identities in Section Identities can be proven to hold using truth tables as follows. In general two propositions are logically equivalent if they take the same value for each set of values of their variables. Thus to see whether or not two propositions are equivalent, we construct truth tables for them and compare to see whether or not they take the same value for each set of values of their variables.

For example consider the commutativity of $\bigvee$:

(P $\bigvee$Q) $\Leftrightarrow$(Q $\bigvee$P).

To prove that this equivalence holds, let us construct a truth table for each of the proposition (P $\bigvee$Q) and (Q $\bigvee$P).

A truth table for (P $\bigvee$Q) is, by the definition of $\bigvee$,

P	Q	(P $\bigvee$Q)
F	F	F
F	T	T
T	F	T
T	T	T

Table 4.8

A truth table for (Q $\bigvee$P) is, by the definition of $\bigvee$,

P	Q	(Q $\bigvee$P)
F	F	F
F	T	T
T	F	T
T	T	T

Table 4.9

As we can see from these tables (P $\bigvee$Q) and (Q $\bigvee$P) take the same value for the same set of value of P and Q. Thus they are (logically) equivalent.

We can also put these two tables into one as follows:

P	Q	(P $\bigvee$Q)	(Q $\bigvee$P)
F	F	F	F
F	T	T	T
T	F	T	T
T	T	T	T

Table 4.10

Using this convention for truth table we can show that the first of De Morgan's Laws also holds.

P	Q	¬(P $\bigvee$Q)	¬P $\bigwedge$¬Q
F	F	T	T
F	T	F	F
T	F	F	F
T	T	F	F

Table 4.11

By comparing the two right columns we can see that ¬(P $\bigvee$Q) and ¬P $\bigwedge$¬Q are equivalent.

4.1.2.4.7 Proof of Implications

1. All the implications in Section Implications can be proven to hold by constructing truth tables and showing that they are always true.

For example consider the first implication "addition": P $\Rightarrow$ (P $\bigvee$ Q).

To prove that this implication holds, let us first construct a truth table for the proposition P $\bigvee$ Q.

P	Q	(P $\bigvee$ Q)
F	F	F
F	T	T
T	F	T
T	T	T

Table 4.12

Then by the definition of $\rightarrow$, we can add a column for P $\rightarrow$ (P $\bigvee$ Q) to obtain the following truth table.

P	Q	(P $\bigvee$ Q)	P $\rightarrow$(P $\bigvee$ Q)
F	F	F	T
F	T	T	T
T	F	T	T
T	T	T	T

Table 4.13

The first row in the rightmost column results since P is false, and the others in that column follow since (P $\bigvee$ Q) is true.

The rightmost column shows that P $\rightarrow$ (P $\bigvee$ Q) is always true.

2. Some of the implications can also be proven by using identities and implications that have already been proven.

For example suppose that the identity "exportation":

[(X $\bigwedge$Y) $\rightarrow$Z] $\Leftrightarrow$[X $\rightarrow$(Y$\rightarrow$Z)] ,

and the implication "hypothetical syllogism":

[(P$\rightarrow$Q) $\bigwedge$(Q$\rightarrow$R)] $\Rightarrow$(P$\rightarrow$R)

have been proven. Then the implication No. 7:

(P$\rightarrow$Q) $\Rightarrow$[(Q$\rightarrow$R)$\rightarrow$(P$\rightarrow$R)]

can be proven by applying the "exportation" to the "hypothetical syllogism" as follows:

Consider (P$\rightarrow$Q) , (Q$\rightarrow$R) , and (P$\rightarrow$R) in the "hypothetical syllogism" as X, Y and Z of the "exportation", respectively.

Then since [(X $\bigwedge$Y)$\rightarrow$Z] $\Leftrightarrow$[X$\rightarrow$(Y$\rightarrow$Z)] implies [(X $\bigwedge$Y)$\rightarrow$Z] $\Rightarrow$[X$\rightarrow$(Y$\rightarrow$Z)] , the implication of No. 7 follows.

Similarly the modus ponens (implication No. 3) can be proven as follows:

Noting that (P$\rightarrow$Q) $\Leftrightarrow$($\neg$P $\bigvee$Q) ,

P $\bigwedge$(P$\rightarrow$Q)

$\Leftrightarrow$P $\bigwedge$($\neg$P $\bigvee$Q)

$\Leftrightarrow$(P $\bigwedge\neg$P) $\bigvee$(P $\bigwedge$Q) — by the distributive law

$\Leftrightarrow$F $\bigvee$(P $\bigwedge$Q)

$\Leftrightarrow$(P $\bigwedge$Q)

$\Rightarrow$Q

Also the exportation (identity No. 20), (P$\rightarrow$(Q$\rightarrow$R)) $\Leftrightarrow$ (P $\bigwedge$Q)$\rightarrow$R) can be proven using identities as follows:

(P$\rightarrow$(Q$\rightarrow$R)) $\Leftrightarrow$ $\neg$P $\bigvee$(Q$\rightarrow$R)

$\Leftrightarrow$ $\neg$P $\bigvee$($\neg$Q $\bigvee$R)

$\Leftrightarrow$ ($\neg$P $\bigvee\neg$Q) $\bigvee$R

$\Leftrightarrow$ $\neg$(P $\bigwedge$Q) $\bigvee$R

$\Leftrightarrow$ (P $\bigwedge$Q)$\rightarrow$R

3. Some of them can be proven by noting that a proposition in an implication can be replaced by an equivalent proposition without affecting its value.

For example by substituting ($\neg$Q$\rightarrow\neg$P) for (P$\rightarrow$Q) , since they are equivalent being contrapositive to each other, modus tollens (the implication No. 4): [(P$\rightarrow$Q) $\bigwedge\neg$Q] $\Rightarrow$ $\neg$P , reduces to the modus ponens: [X $\bigwedge$(X$\rightarrow$Y)] $\Rightarrow$Y. Hence if the modus ponens and the "contrapositive" in the "Identities" have been proven, then the modus tollens follows from them.

4.1.3 Predicate Logic

4.1.3.1 Introduction to Predicate Logic

The propositional logic is not powerful enough to represent all types of assertions that are used in computer science and mathematics, or to express certain types of relationship between propositions such as equivalence.

For example, the assertion "x is greater than 1", where x is a variable, is not a proposition because you can not tell whether it is true or false unless you know the value of x. Thus the propositional logic can not deal with such sentences. However, such assertions appear quite often in mathematics and we want to do inferencing on those assertions.

Also the pattern involved in the following logical equivalences can not be captured by the propositional logic:

"Not all birds fly" is equivalent to "Some birds don't fly".

"Not all integers are even" is equivalent to "Some integers are not even".

"Not all cars are expensive" is equivalent to "Some cars are not expensive",

... .

Each of those propositions is treated independently of the others in propositional logic. For example, if P represents "Not all birds fly" and Q represents "Some integers are not even", then there is no mechanism in propositional logic to find out tha P is equivalent to Q. Hence to be used in inferencing, each of these equivalences must be listed individually rather than dealing with a general formula that covers all these equivalences collectively and instantiating it as they become necessary, if only propositional logic is used.

Thus we need more powerful logic to deal with these and other problems. The predicate logic is one of such logic and it addresses these issues among others.

4.1.3.2 Well Formed Formula (Wff) of Predicate Logic

4.1.3.2.1 Predicate

To cope with deficiencies of propositional logic we introduce two new features: predicates and quantifiers.

A predicate is a verb phrase template that describes a property of objects, or a relationship among objects represented by the variables.

For example, the sentences "The car Tom is driving is blue", "The sky is blue", and "The cover of this book is blue" come from the template "is blue" by placing an appropriate noun/noun phrase in front of it. The phrase "is blue" is a predicate and it describes the property of being blue. Predicates are often given a name. For example any of "is_blue", "Blue" or "B" can be used to represent the predicate "is blue" among others. If we adopt B as the name for the predicate "is_blue", sentences that assert an object is blue can be represented as "B(x)", where x represents an arbitrary object. B(x) reads as "x is blue".

Similarly the sentences "John gives the book to Mary", "Jim gives a loaf of bread to Tom", and "Jane gives a lecture to Mary" are obtained by substituting an appropriate object for variables x, y, and z in the sentence "x gives y to z". The template "... gives ... to ..." is a predicate and it describes a relationship among three objects. This predicate can be represented by Give(x, y, z) or G(x, y, z), for example.

Note: The sentence "John gives the book to Mary" can also be represented by another predicate such as "gives a book to". Thus if we use B(x, y) to denote this predicate, "John gives the book to Mary" becomes B(John, Mary). In that case, the other sentences, "Jim gives a loaf of bread to Tom", and "Jane gives a lecture to Mary", must be expressed with other predicates.

4.1.3.2.2 Quantification — Forming Propositions from Predicates

A predicate with variables is not a proposition. For example, the statement $x > 1$ with variable x over the universe of real numbers is neither true nor false since we don't know what x is. It can be true or false depending on the value of x.

For $x > 1$ to be a proposition either we substitute a specific number for x or change it to something like "There is a number x for which $x > 1$ holds", or "For every number x, $x > 1$ holds".

More generally, a predicate with variables (called an atomic formula) can be made a proposition by applying one of the following two operations to each of its variables:

1. assign a value to the variable
2. quantify the variable using a quantifier (see below).

For example, $x > 1$ becomes $3 > 1$ if 3 is assigned to x, and it becomes a true statement, hence a proposition.

In general, a quantification is performed on formulas of predicate logic (called wff), such as $x > 1$ or P(x), by using quantifiers on variables. There are two types of quantifiers: universal quantifier and existential quantifier.

The universal quantifier turns, for example, the statement $x > 1$ to "for every object x in the universe, $x > 1$", which is expressed as "$\forall x\ x > 1$". This new statement is true or false in the universe of discourse. Hence it is a proposition once the universe is specified.

Similarly the existential quantifier turns, for example, the statement x > 1 to "for some object x in the universe, x > 1", which is expressed as "∃x x > 1." Again, it is true or false in the universe of discourse, and hence it is a proposition once the universe is specified.

4.1.3.2.2.1 Universe of Discourse

The universe of discourse, also called universe, is the set of objects of interest. The propositions in the predicate logic are statements on objects of a universe. The universe is thus the domain of the (individual) variables. It can be the set of real numbers, the set of integers, the set of all cars on a parking lot, the set of all students in a classroom etc. The universe is often left implicit in practice. But it should be obvious from the context.

4.1.3.2.2.2 The Universal Quantifier

The expression: ∀x P(x), denotes the universal quantification of the atomic formula P(x). Translated into the English language, the expression is understood as: "For all x, P(x) holds", "for each x, P(x) holds" or "for every x, P(x) holds". ∀is called the universal quantifier, and ∀x means all the objects x in the universe. If this is followed by P(x) then the meaning is that P(x) is true for every object x in the universe. For example, "All cars have wheels" could be transformed into the propositional form, ∀x P(x), where:

- P(x) is the predicate denoting: x has wheels, and
- the universe of discourse is only populated by cars.

4.1.3.2.2.3 Universal Quantifier and Connective AND

If all the elements in the universe of discourse can be listed then the universal quantification ∀x P(x) is equivalent to the conjunction: P(x1)) ⋀P(x2) ⋀P(x3) ⋀... ⋀P(xn) . For example, in the above example of ∀x P(x), if we knew that there were only 4 cars in our universe of discourse (c1, c2, c3 and c4) then we could also translate the statement as: P(c1) ⋀P(c2) ⋀P(c3) ⋀P(c4) .

4.1.3.2.2.4 The Existential Quantifier

The expression: ∃xP(x), denotes the existential quantification of P(x). Translated into the English language, the expression could also be understood as: "There exists an x such that P(x)" or "There is at least one x such that P(x)" ∃ is called the existential quantifier, and ∃x means at least one object x in the universe. If this is followed by P(x) then the meaning is that P(x) is true for at least one object x of the universe. For example, "Someone loves you" could be transformed into the propositional form, ∃x P(x), where:

- P(x) is the predicate meaning: x loves you,
- The universe of discourse contains (but is not limited to) all living creatures.

4.1.3.2.2.5 Existential Quantifier and Connective OR

If all the elements in the universe of discourse can be listed, then the existential quantification ∃xP(x) is equivalent to the disjunction: P(x1) ⋁P(x2) ⋁P(x3) ⋁... ⋁P(xn).

For example, in the above example of ∃x P(x), if we knew that there were only 5 living creatures in our universe of discourse (say: me, he, she, rex and fluff), then we could also write the statement as: P(me)⋁P(he) ⋁P(she) ⋁P(rex) ⋁P(fluff)

An appearance of a variable in a wff is said to be bound if either a specific value is assigned to it or it is quantified. If an appearance of a variable is not bound, it is called free. The extent of the application (effect) of a quantifier, called the scope of the quantifier, is indicated by square brackets [ ]. If there are no square brackets, then the scope is understood to be the smallest wff following the quantification.

For example, in $\exists$x P(x, y), the variable x is bound while y is free. In $\forall$x [$\exists$y P(x, y) $\bigvee$Q(x, y)], x and the y in P(x, y) are bound, while y in Q(x, y) is free, because the scope of $\exists$y is P(x, y). The scope of $\forall$x is [$\exists$y P(x, y) $\bigvee$Q(x, y)] .

4.1.3.2.2.6 How to read quantified formulas

When reading quantified formulas in English, read them from left to right. $\forall$x can be read as "for every object x in the universe the following holds" and $\exists$x can be read as "there exists an object x in the universe which satisfies the following" or "for some object x in the universe the following holds". Those do not necessarily give us good English expressions. But they are where we can start. Get the correct reading first then polish your English without changing the truth values.

For example, let the universe be the set of airplanes and let F(x, y) denote "x flies faster than y". Then

$\forall$x$\forall$y F(x, y) can be translated initially as "For every airplane x the following holds: x is faster than every (any) airplane y". In simpler English it means "Every airplane is faster than every airplane (including itself !)".

$\forall$x$\exists$ y F(x, y) can be read initially as "For every airplane x the following holds: for some airplane y, x is faster than y". In simpler English it means "Every airplane is faster than some airplane".

$\exists$x$\forall$y F(x, y) represents "There exist an airplane x which satisfies the following: (or such that) for every airplane y, x is faster than y". In simpler English it says "There is an airplane which is faster than every airplane" or "Some airplane is faster than every airplane".

$\exists$x$\exists$ y F(x, y) reads "For some airplane x there exists an airplane y such that x is faster than y", which means "Some airplane is faster than some airplane".

4.1.3.2.2.7 Order of Application of Quantifiers

When more than one variables are quantified in a wff such as $\exists$y $\forall$x P(x, y), they are applied from the inside, that is, the one closest to the atomic formula is applied first. Thus $\exists$y $\forall$x P(x, y) reads $\exists$y [$\forall$x P(x, y)] , and we say "there exists an y such that for every x, P(x, y) holds" or "for some y, P(x, y) holds for every x".

The positions of the same type of quantifiers can be switched without affecting the truth value as long as there are no quantifiers of the other type between the ones to be interchanged.

For example $\exists$x $\exists$y $\exists$z P(x, y , z) is equivalent to $\exists$y $\exists$x $\exists$z P(x, y , z), $\exists$z $\exists$y $\exists$x P(x, y,z), etc. It is the same for the universal quantifier.

However, the positions of different types of quantifiers can not be switched.

For example $\forall$x $\exists$y P(x, y) is not equivalent to $\exists$y $\forall$x P(x, y). For let P(x, y) represent $x < y$ for the set of numbers as the universe, for example. Then $\forall$x $\exists$y P(x, y) reads "for every number x, there is a number y that is greater than x", which is true, while $\exists$y $\forall$x P(x, y) reads "there is a number that is greater than every (any) number", which is not true.

4.1.3.2.3 Well-Formed Formula for First Order Predicate Logic

Not all strings can represent propositions of the predicate logic. Those which produce a proposition when their symbols are interpreted must follow the rules given below, and they are called wffs (well-formed formulas) of the first order predicate logic.

4.1.3.2.3.1 Rules for constructing Wffs

A predicate name followed by a list of variables such as P(x, y), where P is a predicate name, and x and y are variables, is called an atomic formula.

Wffs are constructed using the following rules:

1. True and False are wffs.

2. Each propositional constant (i.e. specific proposition), and each propositional variable (i.e. a variable representing propositions) are wffs.

3. Each atomic formula (i.e. a specific predicate with variables) is a wff.

4. If A, B, and C are wffs, then so are ¬A, (A $\bigwedge$B), (A $\bigvee$ B), (A$\rightarrow$B), and (A $\leftrightarrow$B).

5. If x is a variable (representing objects of the universe of discourse), and A is a wff, then so are $\forall$x A and $\exists$x A.

(Note: More generally, arguments of predicates are something called a term. Also variables representing predicate names (called predicate variables) with a list of variables can form atomic formulas. But we do not get into that here.)

For example, "The capital of Virginia is Richmond." is a specific proposition. Hence it is a wff by Rule 2.

Let B be a predicate name representing "being blue" and let x be a variable. Then B(x) is an atomic formula meaning "x is blue". Thus it is a wff by Rule 3. above. By applying Rule 5. to B(x), $\forall$xB(x) is a wff and so is $\exists$xB(x). Then by applying Rule 4. to them $\forall$x B(x) $\bigwedge\exists$x B(x) is seen to be a wff. Similarly if R is a predicate name representing "being round". Then R(x) is an atomic formula. Hence it is a wff. By applying Rule 4 to B(x) and R(x), a wff B(x) $\bigwedge$R(x) is obtained.

In this manner, larger and more complex wffs can be constructed following the rules given above.

Note, however, that strings that can not be constructed by using those rules are not wffs. For example, $\forall$xB(x)R(x), and B($\exists$x) are NOT wffs, NOR are B(R(x)), and B($\exists$x R(x)) .

One way to check whether or not an expression is a wff is to try to state it in English.

If you can translate it into a correct English sentence, then it is a wff.

More examples: To express the fact that Tom is taller than John, we can use the atomic formula taller(Tom, John), which is a wff. This wff can also be part of some compound statement such as taller(Tom, John) $\bigwedge$ ¬taller(John, Tom), which is also a wff.

If x is a variable representing people in the world, then taller(x,Tom), $\forall$x taller(x,Tom), $\exists$x taller(x,Tom), $\exists$x $\forall$y taller(x,y) are all wffs among others.

However, taller($\exists$ x,John) and taller(Tom $\bigwedge$ Mary, Jim), for example, are NOT wffs.

4.1.3.3 From Wff to Proposition

4.1.3.3.1 Interpretation

A wff is, in general, not a proposition. For example, consider the wff $\forall$x P(x). Assume that P(x) means that x is non-negative (greater than or equal to 0). This wff is true if the universe is the set {1, 3, 5}, the set {2, 4, 6} or the set of natural numbers, for example, but it is not true if the universe is the set {-1, 3, 5}, or the set of integers, for example. Further more the wff $\forall$x Q(x, y), where Q(x, y) means x is greater than y, for the universe {1, 3, 5} may be true or false depending on the value of y.

As one can see from these examples, the truth value of a wff is determined by the universe, specific predicates assigned to the predicate variables such as P and Q, and the values assigned to the free variables. The specification of the universe and predicates, and an assignment of a value to each free variable in a wff is called an interpretation for the wff.

For example, specifying the set {1, 3, 5} as the universe and assigning 0 to the variable y, for example, is an interpretation for the wff $\forall$x Q(x, y), where Q(x, y) means x is greater than y. $\forall$x Q(x, y) with that interpretation reads, for example, "Every number in the set {1, 3, 5} is greater than 0".

As can be seen from the above example, a wff becomes a proposition when it is given an interpretation.

There are, however, wffs which are always true or always false under any interpretation. Those and related concepts are discussed below.

4.1.3.3.2 Satisfiable, Unsatisfiable and Valid Wffs

A wff is said to be satisfiable if there exists an interpretation that makes it true, that is if there are a universe, specific predicates assigned to the predicate variables, and an assignment of values to the free variables that make the wff true.

For example, $\forall x\ N(x)$, where N(x) means that x is non-negative, is satisfiable. For if the universe is the set of natural numbers, the assertion $\forall x\ N(x)$ is true, because all natural numbers are non-negative. Similarly $\exists x\ N(x)$ is also satisfiable.

However, $\forall x\ [N(x) \bigwedge \neg N(x)]$ is not satisfiable because it can never be true. A wff is called invalid or unsatisfiable, if there is no interpretation that makes it true.

A wff is valid if it is true for every interpretation*. For example, the wff $\forall x\ P(x) \bigvee \exists x\ \neg P(x)$ is valid for any predicate name P , because $\exists x\ \neg P(x)$ is the negation of $\forall x\ P(x)$. However, the wff $\forall x\ N(x)$ is satisfiable but not valid.

Note that a wff is not valid iff it is unsatisfiable for a valid wff is equivalent to true. Hence its negation is false.

4.1.3.3.3 Equivalence

Two wffs W1 and W2 are equivalent if and only if W1 $\leftrightarrow$W2 is valid, that is if and only if W1 $\leftrightarrow$W2 is true for all interpretations.

For example $\forall x\ P(x)$ and $\neg\exists x\ \neg P(x)$ are equivalent for any predicate name P. So are $\forall x\ [\ P(x) \bigwedge Q(x)\ ]$ and $[\ \forall x\ P(x) \bigwedge \forall x\ Q(x)\ ]$ for any predicate names P and Q .

4.1.3.4 Transcribing English to Predicate Logic wffs

English sentences appearing in logical reasoning can be expressed as a wff. This makes the expressions compact and precise. It thus eliminates possibilities of misinterpretation of sentences. The use of symbolic logic also makes reasoning formal and mechanical, contributing to the simplification of the reasoning and making it less prone to errors.

Transcribing English sentences into wffs is sometimes a non-trivial task. In this course we are concerned with the transcription using given predicate symbols and the universe. To transcribe a proposition stated in English using a given set of predicate symbols, first restate in English the proposition using the predicates, connectives, and quantifiers. Then replace the English phrases with the corresponding symbols.

Example: Given the sentence "Not every integer is even", the predicate "E(x)" meaning x is even, and that the universe is the set of integers, first restate it as "It is not the case that every integer is even" or "It is not the case that for every object x in the universe, x is even."

Then "it is not the case" can be represented by the connective "$\neg$", "every object x in the universe" by "$\forall$ x", and "x is even" by E(x).

Thus altogether wff becomes $\neg\forall x\ E(x)$.

This given sentence can also be interpreted as "Some integers are not even". Then it can be restated as "For some object x in the universe, x is not integer". Then it becomes $\exists x\ \neg E(x)$.

More examples: A few more sentences with corresponding wffs are given below. The universe is assumed to be the set of integers, E(x) represents x is even, and O(x), x is odd.

"Some integers are even and some are odd" can be translated as

$\exists x\ E(x) \bigwedge \exists x\ O(x)$

"No integer is even" can go to

$\forall x\ \neg E(x)$

"If an integer is not even, then it is odd" becomes

$\forall x\ [\ \neg E(x) \rightarrow O(x)]$

"2 is even" is

E(2)

More difficult translation: In these translations, properties and relationships are mentioned for certain type of elements in the universe such as relationships between integers in the universe of numbers rather

than the universe of integers. In such a case the element type is specified as a precondition using if_then construct.

Examples: In the examples that follow the universe is the set of numbers including real numbers, and complex numbers. I(x), E(x) and O(x) representing "x is an integer", "x is even", and "x is odd", respectively.

"All integers are even" is transcribed as

$\forall x$ [I(x) $\rightarrow$E(x)]

It is first restated as "For every object in the universe (meaning for every numnber in this case) if it is integer, then it is even". Here we are interested in not any arbitrary object(number) but a specific type of objects, that is integers. But if we write $\forall x$ it means "for any object in the universe". So we must say "For any object, if it is integer .." to narrow it down to integers.

"Some integers are odd" can be restated as "There are objects that are integers and odd", which is expressed as

$\exists x$ [I(x) $\bigwedge$E(x)]

"A number is even only if it is integer" becomes

$\forall x$ [E(x)$\rightarrow$I(x)]

"Only integers are even" is equivalent to "If it is even, then it is integer". Thus it is translated to

$\forall x$ [E(x)$\rightarrow$I(x)]

4.1.3.5 Reasoning with Predicate Logic

4.1.3.5.1 Reasoning

Predicate logic is more powerful than propositional logic. It allows one to reason about properties and relationships of individual objects. In predicate logic, one can use some additional inference rules, which are discussed below, as well as those for propositional logic such as the equivalences, implications and inference rules.

The following four rules describe when and how the universal and existential quantifiers can be added to or deleted from an assertion.

1. Universal Instantiation: $\forall x$ P(x)

P(c)

where c is some arbitrary element of the universe.

2. Universal Generalization:

P(c)

$\forall x$ P(x)

where P(c) holds for every element c of the universe of discourse.

3. Existential Instantiation:

$\exists x$ P(x)

P(c)

where c is some element of the universe of discourse. It is not arbitrary but must be one for which P(c) is true.

4. Existential Generalization:

P(c)

$\exists x$ P(x)

where c is an element of the universe.

4.1.3.5.1.1 Negation of Quantified Statement

Another important inference rule is the following:

$\neg\exists x\ P(x) \Leftrightarrow \forall x\ \neg P(x)$

This, for example, shows that if P(x) represents x is happy and the universe is the set of people, then "There does not exist a person who is happy" is equivalent to "Everyone is not happy".

Thus the left side can be substituted for the right side whenever necessary in reasoning and vice versa.

4.1.3.5.1.2 Example

As an example of inference using these rules, let us consider the following reasoning:

A check is void if it has not been cashed for 30 days. This check has not been cashed for 30 days. Therefore this check is void. You can not cash a check which is void. Therefore you can not cash this check. We now have a check which can not be cashed.

This can be put into symbolic form using the following predicates assuming the universe is the set of all objects:

C(x): x is a check.
T(x): x has been cashed within 30 days.
V(x): x is void.
S(x): x can be cashed.
This_check represents a specific object in the universe which corresponds to "this check".

$\forall x\ [\ [\ C(x) \bigwedge \neg T(x)\ ] \rightarrow V(x)\ ]$
$\neg T(\ This\_check\ )$

$V(\ This\_check\ )$
$\forall x\ [\ [\ C(x) \bigwedge V(x)\ ] \rightarrow \neg S(x)\ ]$

$\neg S(\ This\_check\ )$

$\exists x\ [\ C(x) \bigwedge \neg S(x)\ ]$.

Here the reasoning proceeds as follows:
From $\forall x\ [\ [\ C(x) \bigwedge \neg T(x)\ ] \rightarrow V(x)\ ]$ by Universal Instantiation
$[\ [\ C(\ This\_check\ ) \bigwedge \neg T(\ This\_check\ )\ ] \rightarrow V(\ This\_check\ )\ ]$
Since This_check is a check and $\neg T(\ This\_check\ )$,
$[\ C(\ This\_check\ ) \bigwedge \neg T(\ This\_check\ )\ ]$ holds.
Hence
$[\ [\ C(\ This\_check\ ) \bigwedge \neg T(\ This\_check\ )\ ] \rightarrow V(\ This\_check\ )\ ]$
$[\ C(\ This\_check\ ) \bigwedge \neg T(\ This\_check\ )\ ]$

$V(\ This\_check\ )$
by Modus Ponens.
Then from $\forall x\ [\ [\ C(x) \bigwedge V(x)\ ] \rightarrow \neg S(x)\ ]$ by Universal Instantiation,
$[\ [\ C(\ This\_check\ ) \bigwedge V(\ This\_check\ )\ ] \rightarrow \neg S(\ This\_check\ )\ ]$
Since $V(\ This\_check\ )$, and $C(\ This\_check\ )$,
$[\ [\ C(\ This\_check\ ) \bigwedge V(\ This\_check\ )\ ] \rightarrow \neg S(\ This\_check\ )\ ]$
$[\ C(\ This\_check\ ) \bigwedge V(\ This\_check\ )\ ]$

$\neg S(\ This\_check\ )$
by Modus Ponens.
Since $C(\ This\_check\ )$ also holds,
$\neg S(\ This\_check\ )$
$C(\ This\_check\ )$

C(This_check) $\bigwedge\neg$S(This_check)
Then by Existential Generalization $\exists$x [C(x) $\bigwedge\neg$S(x)] .

4.1.3.5.2 Quantifiers and Connectives

4.1.3.5.2.1 Quantifiers and Connectives 1

There are following four important relationships between quantifiers and connectives. They are used frequently in reasoning.

1. $\forall$x [P(x) $\bigwedge$Q(x)] $\Leftrightarrow$[$\forall$x P(x) $\bigwedge\forall$x Q(x)]
2. [$\forall$x P(x) $\bigvee\forall$x Q(x)] $\Rightarrow\forall$x [P(x) $\bigvee$Q(x)]
3. $\exists$x [P(x) $\bigvee$Q(x)] $\Leftrightarrow$[$\exists$x P(x) $\bigvee\exists$x Q(x)]
4. $\exists$x [P(x) $\bigwedge$Q(x)] $\Rightarrow$[$\exists$x P(x) $\bigwedge\exists$x Q(x)]

Let us see what these mean with examples.

Let P(x) represent "x is rich", and Q(x) "x is happy", and let the universe be a set of three people. Also let LHS (RHS) denote the left (right) hand side of the implication or equivalence. Then

1. $\forall$x [P(x) $\bigwedge$Q(x)] $\Leftrightarrow$[$\forall$x P(x) $\bigwedge\forall$x Q(x)] can be illustrated as in Figure 1:

Figure 4.1

That is, LHS says everyone is rich and happy, and RHS says everyone is rich and everyone is happy. Thus LHS and RHS describe the same situation. That is, LHS is true if and only if RHS is true.

2. [$\forall$x P(x) $\bigvee\forall$x Q(x)] $\Rightarrow\forall$x [P(x) $\bigvee$Q(x)] for the same interpretation as 1. above can be shown in Figure 2:

Figure 4.2

That is, LHS says everyone is rich or everyone is happy, and RHS says everyone is rich or happy. Thus if LHS is true, then RHS is certainly true. However on RHS it can happen that two people are rich but the third is not rich but happy. In that case LHS is not true while RHS is true. Thus RHS does not necessarily imply LHS.

3. $\exists$x [P(x) $\bigvee$Q(x)] $\Leftrightarrow$[$\exists$x P(x) $\bigvee\exists$x Q(x)] , again for the same example, can be shown in Figure 3:

Figure 4.3

LHS says someone is rich or happy, and RHS says someone is rich or someone is happy. Thus clearly LHS implies RHS. Also if someone is rich then that person is certainly rich or happy. Thus RHS implies LHS.

4. $\exists x$ [P(x) $\bigwedge$Q(x)] $\Rightarrow$[$\exists x$ P(x) $\bigwedge\exists x$ Q(x)] , for the same example, can be shown in Figure 4:

Figure 4.4

LHS say someone is rich and happy. Hence there is someone who is rich and there is someone who is happy. Hence LHS implies RHS. However, since RHS can be true without anyone being rich and happy at the same time, RHS does not necessarily imply LHS.

4.1.3.5.2.2 Quantifiers and Connectives 2

If a wff (Q below) in the scope of a quantifier does not have the variable (x below) that is quantified by that quantifier, then that wff can be taken out of the scope of that quantifier. That is,

1. $\forall x$ [P(x) $\bigwedge$Q] $\Leftrightarrow$[$\forall x$ P(x) $\bigwedge$Q]
2. [$\forall x$ P(x) $\bigvee$Q] $\Leftrightarrow\forall x$ [P(x) $\bigvee$Q]
3. $\exists x$ [P(x) $\bigvee$Q] $\Leftrightarrow$[$\exists x$ P(x) $\bigvee$Q]
4. $\exists x$ [P(x) $\bigwedge$Q] $\Leftrightarrow$[$\exists x$ P(x) $\bigwedge$Q] ,

where Q in all these formulas DO NOT have the variable x .

Note: When implication $\rightarrow$ and/or equivalence $\leftrightarrow$ are involved, you can not necessarily take Q outside the scope. To see what happens, express $\rightarrow$ and $\leftrightarrow$ using $\bigvee$and $\bigwedge$, and apply the above formulas. For example $\forall x$ [P(x)$\rightarrow$Q] is NOT equivalent to $\forall x$ P(x)$\rightarrow$Q . Rather it is equivalent to $\exists x$ P(x)$\rightarrow$Q. Further details are left as an exercise.

4.1.4 Questions and Exercises

1. Which of the following sentences is a proposition?
   a. Every one is happy.
   b. If it snows, then schools are closed in Norfolk, VA.
   c. x + 2 is positive
   d. Take an umbrella with you.
   e. I suggest that you take an umbrella with you

2. Which of the following tables is a truth table?
Z below represents a proposition involving P and Q.

Table 1		
P	Q	Proposition Z
F	F	F
T	F	T
T	T	T
T	F	T

Table 4.14

Table 2		
P	Q	Proposition Z
F	F	F
T	F	T
T	T	F

Table 4.15

Table 3		
P	Q	Proposition Z
F	F	F
F	T	T
T	F	T
T	T	T

Table 4.16

Table 4	
P	Proposition Z
F	F
F	T
T	F

Table 4.17

3. Indicate which of the following statements are correct and which ones are incorrect.
a. If P is True and Q is False, then P$\bigwedge$Q is True.
b. If P is False and Q is True, then P $\rightarrow$ Q is True.
c. If P is False and Q is False, then P $\leftrightarrow$ Q is False
d. If P is True and Q is False, then P $\bigvee$ Q is True.

e. If P is True and Q is False, then ¬[P⋀Q] is False

4. Indicate which of the following expressions are propositions and which are not.

a. P⋀¬Q.

b. [[P ⋁ Q] → [Q ⋀ R]]

c. [¬[P ↔ ⋀ Q] ⋁ Q]

d. [¬¬P ⋁ Q]

e. [[Q ⋁ R][P ⋀ Q]]

5. Indicate which of the following converses and contrapositives are correct and which are not.

a. If it snows, the schools will be closed.

Converse: If the schools are closed, it snows.

Contrapositive: If the schools are not closed, it does not snow.

b. If I work all night, I can finish this project.

Converse: If I cannot finish this project, I work all night.

Contrapositive: If I can finish this project, I don't work all night.

c. I eat spicy food, only if it upsets my stomach.

Converse: If I eat spicy food, it upsets my stomach.

Contrapositive: If I don't eat spicy food, it doesn't upset my stomach.

6. Which of the following pairs of propositions are logically equivalent?

a. (1) You get promoted only if you have worked hard.

(2) If you have worked hard, you get promoted

b. (1) To get promoted you must work hard.

(2) If you work hard, then you get promoted

c. (1) Whenever there is a noreaster, the beach erodes

(2) If there is a noreaster, the beach erodes.

d. (1) I will stay home, if it snows tonight.

(2) If it snows tonight, I stay home.

7. Indicate which of the following sentences are translated correctly.

Let S represent "It is snowing", F represent "It is below freezing" and G represent "I go outside".

a. "If it is snowing or below freezing, then I don't go outside."

translates to (S ⋁ F) → ¬G

b. "I go outside only if it is neither snowing nor below freezing."

translates to (¬S ⋀ ¬F) → G

c. "Whenever I go outside, it is snowing."

translates to S → G

d. "It is either snowing or below freezing."

translates to S ⋁ F

8. For each of the following propositions, indicate what they are (Tautology, Contingency or Contraction).

a. P → P

b. ¬P → P

c. [[P → Q] ⋀ P] ⋀ ¬Q

d. [P ⋁ [P⋀ Q] → P

e. [P ⋀ [P⋁ Q] ↔ ¬P

10. Indicate which of the following statements are correct and which are not.

a. [R ⋀ ¬S] ↔ [¬S ⋀ R]

b. ¬[P ⋁ [Q ⋀ R]] ↔ [¬P ⋀ ¬[Q ⋀ R]]

c. [[P ⋀ S] ⋁ R] ↔ [[P ⋀ R] ⋁ S]

d. [¬¬P ⋁ Q] ↔ [P → Q]

e. [[Q ⋁ R] ⋀ ¬[R ⋀ Q]] ↔ [Q ⋁ R]

11. Indicate which of the following statements are correct and which are not. If it is correct, what implications are used?

a. If it snows, the schools will be closed. It is snowing.

Therefore, the school is closed.
b. Tom is healthy and (Tom is) happy.
Therefore, Tom is happy
c. John will work at a software company this summer.
Therefore, this summer John will work at a software company and a grocery store.
d. If I work all night, I can finish this project.
But I did not work all night. Therefore, I did not finish the project.
e. If I eat spicy food, it upsets my stomach. If my stomach is upset, I get a bad a dream.
Therefore, if I eat spicy food, I get a bad dream.
12. Indicate which of the following statements are correct and which are not.
Let G(x,y) represent the predicate $x > y$.
a. G(6, 13) means 13 is greater than 6.
b. G(2, 0) is true.
c. G(7, 1) means 7 is greater than 1.
d. "4 is less than 5) can be represented by G(5,4).
13. Indicate which of the following statements are correct and which are not.
Let E(x) mean x is even and G(x,y) mean $x > y$. Let the universe be the set of naturals.
a. $\forall x \exists y$ G(y, x) is true, but $\exists x \forall y$ G(y, x) is false.
b. $\exists y$ E(x) is true.
c. $\forall x \forall y$ G(x, y) is true.
d. $\forall x$ G($\exists y$, x) is a proposition.
14. Indicate which of the following statements are correct and which are not.
a. $\exists x$ [P(x, y) $\forall x \bigwedge$ Q(x,y)] is a wff.
b. $\forall x$ [P(x) $\rightarrow \forall y$ [Q(y) $\rightarrow \exists z$ R(z)]] is a wff.
c. $2 > 1 \bigwedge 3 < 5$ is a wff.
15. Indicate which of the following statements are correct and which are not.
Let P(x) mean x is happy.
a. {Tom} is an interpretation for [$\exists x$ P(x) $\bigwedge$ P(y)].
b. $\forall x$ P(x) is unsatisfiable.
c. {P(Tom) $\bigvee \exists x \neg$P(x)] is valid.
d. $\forall x$ P(x) is equivalent to $\forall y$ P(y)
16. Indicate which of the following statements are correct and which are not.
Let H(x) mean x is happy.
Let the universe be the set of people
a. "If everyone is happy, then Tom is happy" translates to
$\forall x$ [H(x) $\rightarrow$ H(Tom)].
b. "There are happy people" translates to
$\exists x$ H(x)
c. "Not everyone is happy" translates to
$\forall x \neg$[H(x)]
d. "Some people are happy and some are not happy" translates to
$\exists x$ [H(x) $\bigwedge \neg$H(x)]
17. Which of the following sentences are propositions? What are the truth values of those that are propositions?
a. Richmond is the capital of Virginia.
b. $2 + 3 = 7$.
c. Open the door.
d. $5 + 7 < 10$.
e. The moon is a satellite of the earth.
f. $\mathbf{x} + 5 = 7$.
g. $\mathbf{x} + 5 > 9$ for every real number **x**.

18. What is the negation of each of the following propositions?
a. Norfolk is the capital of Virginia.
b. Food is not expensive in the United States.
c. 3 + 5 = 7.
d. The summer in Illinois is hot and sunny.
19. Let **p** and **q** be the propositions
p: Your car is out of gas.
q: You can't drive your car.
Write the following propositions using **p** and **q** and logical connectives.
a. Your car is not out of gas.
b. You can't drive your car if it is out of gas.
c. Your car is not out of gas if you can drive it.
d. If you can't drive your car then it is out of gas.
20. Determine whether each of the following implications is true or false.
a. Your car is not out of gas.
b. If 0.5 is an integer, then 1 + 0.5 = 3.
c. If cars can fly, then 1 + 1 = 3.
d. If 5 > 2 then pigs can fly.
e. If 3*5 = 15 then 1 + 2 = 3.
21. State the converse and contrapositive of each of the following implications.
a. If it snows today, I will stay home.
b. We play the game if it is sunny.
c. If a positive integer is a prime then it has no divisors other than 1 and itself.
22. Construct a truth table for each of the following compound propositions.
a. $\mathbf{p} \bigwedge \neg \mathbf{p}$
b. $(\mathbf{p} \bigvee \neg \mathbf{q}) \rightarrow \mathbf{q}$
c. $(\mathbf{p} \rightarrow \mathbf{q}) \leftrightarrow (\neg \mathbf{q} \rightarrow \neg \mathbf{p})$
23. Write each of the following statements in the form "if **p**, then **q**" in English. (Hint: Refer to the list of common ways to express implications listed in this section.)
a. The newspaper will not come if there is an inch of snow on the street.
b. It snows whenever the wind blows from the northeast.
c. That prices go up implies that supply will be plentiful.
d. It is necessary to read the textbook to understand the materials of this course.
e. For a number to be divisible by 3, it is sufficient that it is the sum of three consecutive integers.
f. Your guarantee is good only if you bought your TV less than 90 days ago.
24. Write each of the following propositions in the form "**p** if and only if **q**" in English.
a. If it is hot outside you drink a lot of water, and if you drink a lot of water it is hot outside.
b. For a program to be readable it is necessary and sufficient that it is well structured.
c. I like fruits only if they are fresh, and fruits are fresh only if I like them.
d. If you eat too much sweets your teeth will decay, and conversely.
e. The store is closed on exactly those days when I want to shop there.
25. Use truth table to verify the following equivalences.
a. $\mathbf{p} \bigwedge$ False $\Leftrightarrow$ False
b. $\mathbf{p} \bigvee$ True $\Leftrightarrow$ True
c. $\mathbf{p} \bigvee \mathbf{p} \Leftrightarrow \mathbf{p}$
26. Use truth tables to verify the distributive law $\mathbf{p} \bigwedge (\mathbf{q} \bigvee \mathbf{r}) \Leftrightarrow (\mathbf{p} \bigwedge \mathbf{q}) \bigvee (\mathbf{p} \bigwedge \mathbf{r})$.
27. Show that each of the following implications is a tautology without using truth tables.
a. $\mathbf{p} \rightarrow (\mathbf{p} \bigvee \mathbf{q})$
b. $(\mathbf{p} \bigwedge \mathbf{q}) \rightarrow (\mathbf{p} \rightarrow \mathbf{q})$
c. $\neg(\mathbf{p} \rightarrow \mathbf{q}) \rightarrow \neg \mathbf{q}$
28. Verify the following equivalences, which are known as the absorption laws.

a. [$\mathbf{p} \bigvee(\mathbf{p} \bigwedge \mathbf{q})$] $\Leftrightarrow \mathbf{p}$
b. [$\mathbf{p} \bigwedge(\mathbf{p} \bigvee \mathbf{q})$] $\Leftrightarrow \mathbf{p}$

29. Find the dual of each of the following propositions.

a. $\mathbf{p} \bigvee \neg \mathbf{q} \bigvee \neg \mathbf{r}$
b. $(\mathbf{p} \bigvee \mathbf{q} \bigvee \mathbf{r}) \bigwedge \mathbf{s}$
c. $(\mathbf{p} \bigwedge \mathrm{F}) \bigvee(\mathbf{q} \bigwedge \mathrm{T})$

30. Find a compound proposition involving the propositions **p**, **q**, and **r** that is true when exactly one of **p**, **q**, and **r** is true and is false otherwise. (**Hint**: Form a disjunction of conjunctions. Include a conjunction for each combination of values for which the proposition is true. Each conjunction should include each of the three propositions or their negations).

31. What rule of inference is used in each of the following arguments?

a. John likes apple pies. Therefore, John likes apple pies or icecream.

b. Mary likes chocolate and icecream. Therefore, Mary likes chocolate.

c. If it snows, then the roads are closed; it snows. Therefore, the roads are closed.

d. If it snows, then the roads are closed; the roads are not closed. Therefore, it does not snow.

e. To go to Tahiti, one must fly or take a boat; there is no seat on any flight to Tahiti this year. Therefore, one must take a boat to go to Tahiti this year.

32. Express the following arguments using the symbols indicated. What rules of inference are used in each of them?

a. If the teens like it, then the sales volume will go up; Either the teens like it or the store will close; The sales volume will not go up. Therefore, the store will close.

Symbols to be used: The teens like it (T). The sales volume will go up (S). The store will close (C).

b. It is not the case that if there is not a lot of sun, then there is enough water, nor is it true that either there is a lot of rain or the crop is good. Therefore, there is not enough water and the crop is not good. Symbols to be used: There is not a lot of sun (S). There is enough water (W). There is a lot of rain (R). The crop is good (C).

c. If flowers are colored, they are always scented; I don't like flowers that are not grown in the open air; All flowers grown in the open air are colored. Therefore, I don't like any flowers that are scentless. Symbols to be used: Flowers are colored (C). Flowers are scented (S). I like flowers (L). Flowers are grown in the open air (O).

d. No animals, except giraffes, are 15 feet or higher; There are no animals in this zoo that belong to anyone but me; I have no animals less than 15 feet high. Therefore, all animals in this zoo are giraffes. Symbols to be used: Animals are giraffes (G). Animals are 15 feet or higher (F). Animals are in the zoo (Z). Animals belong to me (M).

e. Bees like red flowers, or my hat is red and bees like hats; However, my hat is not red, or bees don't like hats but they like red flowers. Therefore bees like red flowers. Symbols to be used: Bees like red flowers (R). My hat is red (H). Bees like hats (L).

33. Let $\mathbf{Q(x, y)}$ denote the statement "**x** is greater than **y**." What are the truth values of the following?

a. $\mathbf{Q}$ (3, 1)
b. $\mathbf{Q}$ (5, 5)
c. $\mathbf{Q}$ (6, -6)
d. $\mathbf{Q}$ (28, 256)

34. Let $\mathbf{P(x)}$ be the statement "**x** is happy," where the universe of discourse for **x** is the set of students. Express each of the following quantifications in English.

a. $\exists \mathbf{x}\ \mathbf{P(x)}$
b. $\forall \mathbf{x}\ \neg \mathbf{P(x)}$
c. $\exists \mathbf{x}\ \neg \mathbf{P(x)}$
d. $\neg \forall \mathbf{x}\ \neg \mathbf{P(x)}$

35. Let $\mathbf{P(x)}$ be the statement " $\mathbf{x} > \mathbf{x}2$." If the universe of discourse is the set of real numbers, what are the truth values of the following?

a. $\mathbf{P(0)}$

b. P(1/2)
c. P(2)
d. P(-1)
e. $\exists$**x P(x)**
f. $\forall$**x P(x)**

36. Suppose that the universe of discourse of the atomic formula **P(x,y)** is {1, 2, 3}. Write out the following propositions using disjunctions and conjunctions.

a. $\exists$**x P(x, 2)**
b. $\forall$**y P(3, y)**
c. $\forall$**x** $\forall$**yP(x, y)**
d. $\exists$**x**$\exists$ **yP(x, y)**
e. $\exists$**x** $\forall$**y P(x, y)**
f. $\forall$**y** $\exists$**x P(x, y)**

37. Let **L(x, y)** be the predicate "**x** likes y," and let the universe of discourse be the set of all people. Use quantifiers to express each of the following statements.

a. Everyone likes everyone.
b. Everyone likes someone.
c. Someone does not like anyone.
d. Everyone likes George.
e. There is someone whom everyone likes.
f. There is no one whom everyone likes.
g. Everyone does not like someone.

38. Let **S(x)** be the predicate "**x** is a student," **B(x)** the predicate "**x** is a book," and **H(x,y)** the predicate "**x** has **y**, " where the universe of discourse is the universe, that is the set of all objects. Use quantifiers to express each of the following statements.

a. Every student has a book.
b. Some student does not have any book.
c. Some student has all the books.
d. Not every student has a book.
e. There is a book which every student has.

39. Let **B(x)**, **E(x)** and **G(x)** be the statements "**x** is a book," "**x** is expensive, "and "**x** is good," respectively. Express each of the following statements using quantifiers; logical connectives; and **B(x)**, **E(x)** and **G(x)**, where the universe of discourse is the set of all objects.

a. No books are expensive.
b. All expensive books are good.
c. No books are good.
d. Does (c) follow from (a) and (b)?

40. Let **G(x)**, **F(x)**, **Z(x)**, and **M(x)** be the statements "**x** is a giraffe," "**x** is 15 feet or higher, ""**x** is in this zoo, "and "**x** belongs to me," respectively. Suppose that the universe of discourse is the set of animals. Express each of the following statements using quantifiers; logical connectives; and **G(x)**, **F(x)**, **Z(x)**, and **M(x)**.

a. No animals, except giraffes, are 15 feet or higher;
b. There are no animals in this zoo that belong to anyone but me;
c. I have no animals less than 15 feet high.
d. Therefore, all animals in this zoo are giraffes.
e. Does (d) follow from (a), (b), and (c)? If not, is there a correct conclusion?

41. Show that the statements $\neg\exists$**x** $\forall$**y P(x, y)** and $\forall$**x** $\exists$**y** $\neg$**P(x, y)** have the same truth value.

42. For each of the following arguments, explain which rules of inference are used for each step. The universe is the set of people.

a. "John, a student in this class, is 16 years old. Everyone who is 16 years old can get a driver's license. Therefore, someone in this class can get a driver's license."

b. "Somebody in this class enjoys hiking. Every person who enjoys hiking also likes biking. Therefore, there is a person in this class who likes biking."

c. "Every student in this class owns a personal computer. Everyone who owns a personal computer can use the Internet. Therefore, John, a student in this class, can use the Internet."

d. "Everyone in this class owns a personal computer. Someone in this class has never used the Internet. Therefore, someone who owns a personal computer has never used the Internet."

43. Determine whether each of the following arguments is valid. If an argument is correct, what rule of inference is being used? If it is not, what fallacy occurs?

a. "If $\mathbf{n}$ is a real number with $\mathbf{n} > 1$, then $\mathbf{n}2 > 1$. Suppose that $\mathbf{n}2 \leq 1$. Then $\mathbf{n} \leq 1$.

b. "If $\mathbf{n}$ is a real number with $\mathbf{n} > 1$, then $\mathbf{n}2 > 1$. Suppose that $\mathbf{n}2 > 1$. Then $\mathbf{n} > 1$.

44. Show that $\exists \mathbf{x}\ \mathbf{P(x)} \bigwedge \exists \mathbf{x}\ \mathbf{Q(x)}$ and $\exists \mathbf{x(P(x)} \bigwedge \mathbf{Q(x))}$ are not logically equivalent.

45. Show that $\forall \mathbf{x}\ \mathbf{P(x)} \bigwedge \exists \mathbf{x}\ \mathbf{Q(x)}$ and $\forall \mathbf{x}\ \exists \mathbf{y(P(x)} \bigwedge \mathbf{Q(y))}$ are equivalent.

Chapter 5

Discrete Structures Set Theory[1]

5.1 Set Theory

5.1.1 Basics

5.1.1.1 Introduction to Set Theory

The concept of set is fundamental to mathematics and computer science. Everything mathematical starts with sets. For example, relationships between two objects are represented as a set of ordered pairs of objects, the concept of ordered pair is defined using sets, natural numbers, which are the basis of other numbers, are also defined using sets, the concept of function, being a special type of relation, is based on sets, and graphs and digraphs consisting of lines and points are described as an ordered pair of sets. Though the concept of set is fundamental to mathematics, it is not defined rigorously here. Instead we rely on everyone's notion of "set" as a collection of objects or a container of objects. In that sense "set" is an undefined concept here. Similarly we say an object "belongs to" or "is a member of" a set without rigorously defining what it means. "An object (element) x belongs to a set A" is symbolically represented by "$x \in A$". It is also assumed that sets have certain (obvious) properties usually associated with a collection of objects such as the union of sets exists, for any pair of sets there is a set that contains them etc.

This approach to set theory is called "naive set theory" as opposed to more rigorous "axiomatic set theory". It was first developed by the German mathematician Georg Cantor at the end of the 19th century. Though the naive set theory is not rigorous, it is simpler and practically all the results we need can be derived within the naive set theory. Thus we shall be following this naive set theory in this course.

5.1.1.2 Representation of Set

A set can be described in a number of different ways. The simplest is to list up all of its members if that is possible. For example {1, 2, 3} is the set of three numbers 1, 2, and 3. { indicates the beginning of the set, and } its end. Every object between them separated by commas is a member of the set. Thus {{1, 2}, {{3}, 2}, 2}, {1 } } is the set of the elements {1, 2}, {{3}, 2} and {1}.

A set can also be described by listing the properties that its members must satisfy. For example, { x| 1 $\leq$x $\leq$2 and x is a real number. } represents the set of real numbers between 1 and 2, and { x| x is the square of an integer and x $\leq$100 } represents the set { 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100 }.

A third way to describe a set is to give a procedure to generate the members of the set. The recursive/inductive definition is an example and it is going to be studied later. In this representation, first, basic elements of the set are presented. Then a method is given to generate elements of the set from known elements of the set. Thirdly a statement is given that excludes undesirable elements (which may be included

[1] This content is available online at <http://cnx.org/content/m15772/1.1/>.

in the set otherwise) from the set. For example the set of natural numbers N can be defined recursively as the set that satisfies the following (1), (2), and (3):

(1) $0 \in N$

(2) For any number x if $x \in N$, then $x + 1 \in N$.

(3) Nothing is in N unless it is obtained from (1) and (2).

Following this definition, the set of natural numbers N can be obtained as follows: First by (1), 0 is put into N.

Then by (2), since 0 is in N, $0 + 1$ (= 1) is in N.

Then by (2) again, $1 + 1$ (= 2) is in N.

Proceeding in this manner all the natural numbers are put into N. Note that if we don't have (3), 0.5, 1.5, 2.5, ... can be included in N, which is not what we want as the set of natural numbers.

5.1.1.3 Basics of Set

Definition (Equality of sets): Two sets are equal if and only if they have the same elements.

More formally, for any sets A and B, A = B if and only if $\forall x [x \in A \leftrightarrow x \in B]$.

Thus for example {1, 2, 3} = {3, 2, 1}, that is the order of elements does not matter, and {1, 2, 3} = {3, 2, 1, 1}, that is duplications do not make any difference for sets.

Definition (Subset): A set A is a subset of a set B if and only if everything in A is also in B.

More formally, for any sets A and B, A is a subset of B, and denoted by $A \subseteq B$, if and only if $\forall x [x \in A \rightarrow x \in B]$.

If $A \subseteq B$, and $A \neq B$, then A is said to be a proper subset of B and it is denoted by $A \subset B$.

For example $\{1, 2\} \subseteq \{3, 2, 1\}$.

Also $\{1, 2\} \subset \{3, 2, 1\}$.

Definition (Cardinality): If a set S has n distinct elements for some natural number n, n is the cardinality (size) of S and S is a finite set. The cardinality of S is denoted by |S|. For example the cardinality of the set {3, 1, 2} is 3.

Definition (Empty set): A set which has no elements is called an empty set. More formally, an empty set, denoted by $\emptyset$, is a set that satisfies the following:

$\forall x\ x \notin \emptyset$, where $\notin$ means "is not in" or "is not a member of".

Note that $\emptyset$ and $\{\emptyset\}$ are different sets. $\{\emptyset\}$ has one element namely $\emptyset$ in it. So $\{\emptyset\}$ is not empty. But $\emptyset$ has nothing in it.

Definition (Universal set): A set which has all the elements in the universe of discourse is called a universal set.

More formally, a universal set, denoted by U, is a set that satisfies the following: $\forall x\ x \in U$.

Three subset relationships involving empty set and universal set are listed below as theorems without proof.

Note that the set A in the next four theorems are arbitrary. So A can be an empty set or universal set.

Theorem 1: For an arbitrary set A $A \subseteq U$.

Theorem 2: For an arbitrary set A $\emptyset \subseteq A$.

Theorem 3: For an arbitrary set A $A \subseteq A$.

Definition (Power set): The set of all subsets of a set A is called the power set of A and denoted by 2^A or P(A) .

For example for A = {1, 2}, P(A) = {$\emptyset$, {1}, {2}, {1, 2} } .

For B = {{1, 2}, {{1}, 2}, $\emptyset$} , P(B) = {$\emptyset$, {{1, 2}}, {{{1}, 2}}, {$\emptyset$}, { {1, 2}, {{1}, 2 }}, { {1, 2}, $\emptyset$}, { {{1}, 2}, $\emptyset$}, {{1, 2}, {{1}, 2}, $\emptyset$} } .

Also P($\emptyset$) = {$\emptyset$} and P({$\emptyset$}) = {$\emptyset$, {$\emptyset$}} .

Theorem 4: For an arbitrary set A, the number of subsets of A is $2^{|A|}$.

5.1.2 Mathematical Reasoning

Mathematical theories are constructed starting with some fundamental assumptions, called axioms, such as "sets exist" and "objects belong to a set" in the case of naive set theory, then proceeding to defining concepts(definitions) such as "equality of sets", and "subset", and establishing their properties and relationships between them in the form of theorems such as "Two sets are equal if and only if each is a subset of the other", which in turn causes introduction of new concepts and establishment of their properties and relationships. Proofs are the arguments for establishing those properties and relationships. At the bottom level these arguments follow the inference rules of propositional and predicate logic, that is the conclusion of the theorem being proved must be derived from its hypotheses, axioms, definitions, and proven theorems using inference rules. However, at the bottom level they become tedious and inefficient as one can easily imagine. Thus in actual proofs short-cuts are taken using already proven theorems, using multiple inference rules in one step without explicitly mentioning them individually, omitting "obvious" proofs, and so on.

Finding a proof is in general an art. There is no single method that works for all cases. However, at this level the most important thing to remember is to know and understand definitions of concepts involved. The next important thing to keep in mind is to look up relevant facts and try to use them. Even if you don't see the entire path to the goal, if you move one step forward from where you are, you get a new perspective and it often gives you some good ideas to pursue. Needless to say that you must not forget the inference rules. It is not a bad idea to review "Problem Solving" we studied earlier here.

There are also some well used and often very useful proof techniques such as trivial proof, vacuous proof, direct proof, proof by contradiction, proving the contrapositive, and proof by induction. These are explained below with proofs of the theorems on subset relation as examples.

Theorem 1: $A \subseteq U$.

Proof: By the definition of $\subseteq$, we need to show that $\forall x\ [x \in A \rightarrow x \in U]$.

For that, what we need is to show that for an arbitrary x, $x \in A \rightarrow x \in U$ holds according to the inference rule "universal generalization".

Since x is an object of the universe of discourse, $x \in U$ is true for any arbitrary object by the Universal Instantiation. Hence $x \in A \rightarrow x \in U$ is true for any arbitrary object x ($p \rightarrow q$ is always true if q is true regardless of what p is). Thus by the Universal Generalization $\forall x\ [x \in A \rightarrow x \in U]$, that is, $A \subseteq U$ by the definition of subset.

We say $p \rightarrow q$ is trivially true if q is true, and this kind of proof (i.e. showing q is true for $p \rightarrow q$ without referring to p) is called a trivial proof.

Theorem 2: $\emptyset \subseteq A$.

Proof: By the definition of $\subseteq$, we need to show that $\forall x\ [x \in \emptyset \rightarrow x \in A]$. For that, what we need is to show that for an arbitrary x, $x \in \emptyset \rightarrow x \in A$ holds. Then apply the Universal Generalization.

Since $\forall x\ x \notin \emptyset$, for any arbitrary x, $x \in \emptyset$ is false by the Universal Instantiation.

Hence $x \in \emptyset \rightarrow x \in A$ is true for any arbitrary x ($p \rightarrow q$ is always true if p is false regardless of what q is). Hence by the Universal Generalization $\forall x\ [x \in \emptyset \rightarrow x \in A]$. Thus $\emptyset \subseteq A$ by the definition of subset.

We say $p \rightarrow q$ is vacuously true if p is false, and this kind of proof (i.e. showing p is false for $p \rightarrow q$) is called a vacuous proof.

Proof Variations for Theorem 2

Theorem 2, like most others, can be proven in a number of other ways. Here we try to prove it in two other ways.

(1) Proof by Contrapositive:

In this method, to prove $p \rightarrow q$ we prove its contrapositive, $\neg q \rightarrow \neg p$, instead. So to prove $x \in \emptyset \rightarrow x \in A$ for an arbitrary x, try to prove $x \notin A \rightarrow x \notin \emptyset$. In this case since $x \notin \emptyset$ is true for any arbitrary x, $x \notin A \rightarrow x \notin \emptyset$ is trivially true. Hence its contrapositive $x \in \emptyset \rightarrow x \in A$ is also true.

(2) Proof by Contradiction:

In this method, to prove p we assume $\neg p$ and derive a contradiction from that. Then since $\neg p$ implies a contradiction, it can not hold true. Hence p must be true.So to prove $\forall x\ [x \in \emptyset \rightarrow x \in A]$ we assume that $\neg \forall x\ [x \in \emptyset \rightarrow x \in A]$. $\neg \forall x\ [x \in \emptyset \rightarrow x \in A]$ is equivalent to $\exists x\ \neg [x \in \emptyset \rightarrow x \in A]$. This in turn is equivalent to $\exists x\ [x \in \emptyset \bigwedge x \notin A]$.

However, $\exists x\ [x \in \emptyset \bigwedge x \notin A]$ implies $\exists x\ [x \in \emptyset]$ by formula 4 of the relationships between the connectives and quantifiers from predicate logic and "simplification" from the implications of propositional logic. But $x \notin \emptyset$ for any x, which contradicts $\exists x\ [x \in \emptyset]$. Hence, $\neg\forall x\ [x \in \emptyset \rightarrow x \in A]$ can not be true. Hence, $\emptyset \subseteq A$.

More Proofs

Theorem 3: $A = B$ iff $A \subseteq B$ and $B \subseteq A$

Proof: By the definition of $A = B$,

$A = B \Leftrightarrow \forall x\ [x \in A \leftrightarrow x \in B]$

$\Leftrightarrow \forall x\ [(x \in A \rightarrow x \in B) \bigwedge (x \in B \rightarrow x \in A)\ ]$

$\Leftrightarrow \forall x\ [x \in A \rightarrow x \in B] \bigwedge \forall x\ [\ (x \in B \rightarrow x \in A)\ ]$

Since $A \subseteq B \Leftrightarrow \forall x\ [\ (x \in A \rightarrow x \in B)\ ]$, this means that $A \subseteq B$ and $B \subseteq A$.

Hence, $A = B$ iff $A \subseteq B$ and $B \subseteq A$.

Theorem 4: $A \subseteq B \bigwedge B \subseteq C \rightarrow A \subseteq C$Proof: $A \subseteq B \bigwedge B \subseteq C$

$\Leftrightarrow \forall x\ [x \in A \rightarrow x \in B] \bigwedge \forall x\ [\ (x \in B \rightarrow x \in C)\ ]$

$\Leftrightarrow \forall x\ [(x \in A \rightarrow x \in B) \bigwedge (x \in B \rightarrow x \in C)\ ]$

Thus for an arbitrary x in the universe, Z, and $x \in B \rightarrow x \in C$ holds.Hence, by hypothetical syllogism $x \in A \rightarrow x \in C$,Hence, $A \subseteq C$.

5.1.3 Set Operations

5.1.3.1 Set Operations

Sets can be combined in a number of different ways to produce another set. Here four basic operations are introduced and their properties are discussed.

Definition (Union): The union of sets A and B, denoted by $A \cup B$, is the set defined as

$A \cup B = \{x \mid x \in A \vee x \in B\}$

Example 1: If $A = \{1, 2, 3\}$ and $B = \{4, 5\}$, then $A \cup B = \{1, 2, 3, 4, 5\}$.

Example 2: If $A = \{1, 2, 3\}$ and $B = \{1, 2, 4, 5\}$, then $A \cup B = \{1, 2, 3, 4, 5\}$.

Note that elements are not repeated in a set.

Definition (Intersection): The intersection of sets A and B, denoted by $A \cap B$, is the set defined as

$A \cap B = \{x \mid x \in A \bigwedge x \in B\}$

Example 3: If $A = \{1, 2, 3\}$ and $B = \{1, 2, 4, 5\}$, then $A \cap B = \{1, 2\}$.

Example 4: If $A = \{1, 2, 3\}$ and $B = \{4, 5\}$, then $A \cap B = \emptyset$.

Definition (Difference): The difference of sets A from B, denoted by A - B, is the set defined as

$A - B = \{x \mid x \in A \bigwedge x \notin B\}$

Example 5: If $A = \{1, 2, 3\}$ and $B = \{1, 2, 4, 5\}$, then $A - B = \{3\}$.

Example 6: If $A = \{1, 2, 3\}$ and $B = \{4, 5\}$, then $A - B = \{1, 2, 3\}$.

Note that in general $A - B \neq B - A$

Definition (Complement): For a set A, the difference U - A, where U is the universe, is called the complement of A and it is denoted by $\overline{A}$.

Thus $\overline{A}$ is the set of everything that is not in A.

The fourth set operation is the Cartesian product. We first define an ordered pair and Cartesian product of two sets using it. Then the Cartesian product of multiple sets is defined using the concept of n-tuple.

Definition (ordered pair):

An ordered pair is a pair of objects with an order associated with them. If objects are represented by x and y, then we write the ordered pair as <x, y>.

Two ordered pairs <a, b> and <c, d> are equal if and only if a = c and b = d. For example the ordered pair <1, 2> is not equal to the ordered pair <2, 1>.

Definition (Cartesian product):

The set of all ordered pairs <a, b>, where a is an element of A and b is an element of B, is called the Cartesian product of A and B and is denoted by $A \times B$.

Example 1: Let $A = \{1, 2, 3\}$ and $B = \{a, b\}$. Then $A \times B = \{<1, a>, <1, b>, <2, a>, <2, b>, <3, a>, <3, b>\}$.

Example 2: For the same A and B as in Example 1,
B ×A = {<a, 1>, <a, 2>, <a, 3>, <b, 1>, <b, 2>, <b, 3>}.
As you can see in these examples, in general, A ×B ≠B ×A unless A = ∅, B = ∅ or A = B.
Note that A × ∅= ∅ × A = ∅ because there is no element in ∅ to form ordered pairs with elements of A.
The concept of Cartesian product can be extended to that of more than two sets. First we are going to define the concept of **ordered n-tuple**.
Definition (ordered n-tuple): An ordered n-tuple is a set of n objects with an order associated with them (rigorous definition to be filled in). If n objects are represented by x1, x2, ..., xn, then we write the ordered n-tuple as <x1, x2, ..., xn> .
Definition (Cartesian product): Let A1, ..., An be n sets. Then the set of all ordered n-tuples <x1, ..., xn> , where xi∈Ai for all i, 1 ≤ i ≤ n , is called the Cartesian product of A1, ..., An, and is denoted by A1 ×... ×An .
Example 3:
Let A = {1, 2}, B = {a, b} and C = {5, 6}. Then A ×B ×C = {<1, a, 5>, <1, a, 6>, <1, b, 5>, <1, b, 6>, <2, a, 5>, <2, a, 6>, <2, b, 5>, <2, b, 6>}.
Definition (equality of n-tuples): Two ordered n-tuples <x1, ..., xn> and <y1, ..., yn> are equal if and only if xi = yi for all i, 1 ≤i ≤n.
For example the ordered 3-tuple <1, 2, 3> is not equal to the ordered n-tuple <2, 3, 1>.

5.1.3.2 Properties of Set Operation

Basic properties of set operations are discussed here. 1 - 6 directly correspond to identities and implications of propositional logic, and 7 - 11 also follow immediately from them as illustrated below.
1. A ∪ ∅ = A
A ∩ U = A
——- Identity Laws
2. A ∪ U = U
A ∩ ∅ = ∅
——- Domination Laws
3. A ∪ A = A
A ∩ A = A
——- Idempotent Laws
4. A ∪ B = B ∪ A
A ∩ B = B ∩A
——- Commutative Laws
5. (A ∪ B) ∪ C = A ∪ (B ∪ C)
(A ∩ B) ∩ C = A ∩ (B ∩ C)
——- Associative Laws
6. A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C)
A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)
——- Distributive Laws
7. If A⊆B and C⊆D, then A ∪ C ⊆ B ∪ D, and A ∩ C ⊆ B ∩ D
8. If A⊆B, then A ∪ B = B and A ∩ B = A
9. A ∪ (B - A) = A ∪ B
10. A ∩ (B - A) = ∅
11. A - (B ∪ C) = (A − B) ∩ (A - C) (cf. $\overline{A \cup B} = \overline{A} \cap \overline{B}$)
A - (B ∩ C) = (A - B) ∪ (A - C) (cf. $\overline{A \cap B} = \overline{A} \cup \overline{B}$)
——- De Morgan's Laws
12. $B = \overline{A}$ if and only if A ∪ B = U and A ∩ B = ∅
13. $\overline{\overline{A}} = A$
Additional properties:
14. A ⊆A ∪B

15. $A \cap B \subseteq A$

The properties 1∼6, and 11 can be proven using equivalences of propositional logic. The others can also be proven similarly by going to logic, though they can be proven also using some of these properties (after those properties are proven, needless to say). Let us prove some of these properties.

Proof for 4: $A \cup B = B \cup A$

We are going to prove this by showing that every element that is in A∪B is also in B∪A and vice versa.

Consider an arbitrary element x. Then by the definition of set union $x \in A \cup B \Leftrightarrow x \in A \vee x \in B$

$\Leftrightarrow x \in A \vee x \in B$ by the commutativity of $\vee$

$\Leftrightarrow x \in B \cup A$ by the definition of set union.

Hence by Universal Generalization, every element is in A ∪B is also in B ∪A.

Hence $A \cup B = B \cup A$.

Note here the correspondence of the commutativity of $\cup$ and that of $\vee$. This correspondence holds not just for the commutativity but also for others.

Furthermore a similar correspondence exists between $\cap$ and $\bigwedge$, and between $\subseteq$ and $\rightarrow$.

Proof for 6: By the definition of the equality of sets, we need to prove that $\forall x\ [x \in A \cup (B \cap C)]$ if and only if $x \in (A \cup B) \cap (A \cup C)$.

For that, considering the Universal Generalization rule, we need to show that for an arbitrary element in the universe x, $x \in A \cup (B \cap C)$ if and only if $x \in (A \cup B) \cap (A \cup C)$.

Here the only if part is going to be proven. The if part can be proven similarly.

$x \in A \cup (B \cap C) \Leftrightarrow x \in A \vee x \in (B \cap C)$ by the definition of $\cup$

$\Leftrightarrow x \in A \vee (x \in B \bigwedge x \in C)$ by the definition of $\cap$

$\Leftrightarrow (x \in A \vee x \in B) \bigwedge (x \in A \vee x \in C)$ by the distribution from the equivalences of propositional logic

$\Leftrightarrow x \in (A \cup B) \bigwedge x \in (A \cup C)$ by the definition of $\cup$.

$\Leftrightarrow x \in (A \cup B) \cap (A \cup C)$ by the definition of $\cap$.

Proof for 8: (a) If $A \subseteq B$ then $A \cup B = B$.

Let x be an arbitrary element in the universe.

Then $x \in A \cup B \Leftrightarrow x \in A \vee x \in B$.

Since $A \subseteq B$, $x \in A \rightarrow x \in B$

Also $x \in B \rightarrow x \in B$

Hence $x \in A \cup B \rightarrow x \in B$.

Hence $A \cup B \subseteq B$

Since $B \subseteq A \cup B$ (use "addition" rule), $A \cup B = B$ follows.

(b) Similarly for $A \cap B = A$.

Alternative proof:

These can also be proven using 8, 14, and 15. For example, (b) can be proven as follows:

First by 15, $A \cap B \subseteq A$.

Then since $A \subseteq A$, and $A \subseteq B$, by 7 $A \cap A \subseteq A \cap B$.

Since $A \cap A = A$ by 3, $A \subseteq A \cap B$.

Proof for 9: Let x be an arbitrary element in the universe.

Then $[x \in A \cup (B - A)] \Leftrightarrow [x \in A \vee (x \in B \bigwedge x \notin A)]$

$\Leftrightarrow [(x \in A \vee x \in B) \bigwedge (x \in A \vee x \notin A)]$

$\Leftrightarrow [(x \in A \vee x \in B) \bigwedge \text{True}]$

$\Leftrightarrow [x \in A \vee x \in B]$

Hence $A \cup (B - A) = A \cup B$.

Alternative proof

This can also proven using set properties as follows.

$A \cup (B - A) = A \cup (B \cap \overline{A})$ by the definition of $(B - A)$.

$= (A \cup B) \cap (A \cup \overline{A})$ by the distribution.

$= (A \cup B) \cap U$

$= (A \cup B)$ by 1.

Proof for 10: Suppose $A \cap (B - A) \neq \emptyset$.

Then there is an element x that is in A ∩ (B - A), i.e.
x ∈ A ∩ (B - A) ⇔ x ∈ A ⋀ x ∈ B − A
⇔ x ∈ A ⋀ (x ∈ B ⋀ x ∉ A)
⇔ (x ∈ A ⋀ x ∉ A) ⋀ x ∈ B
⇔ False
Hence A ∩ (B - A) ≠ ∅ does not hold.
Hence A ∩ (B - A) = ∅.
This can also be proven in the similar manner to 9 above.
Proof for 11: Let x be an arbitrary element in the universe.
Then x ∈ A - (B ∪ C) ⇔ x ∈ A ⋀ x ∉ B ∪ C
⇔ x ∈ A ⋀ ¬(x ∈ B ⋁ x ∈C)
⇔ x ∈ A ⋀ (x ∉ B ⋀ x ∉C)
⇔ (x ∈ A ⋀ x ∉ B) ⋀ x ∉C
⇔ (x ∈ A ⋀ x ∉ B) ⋀ (x ∈ A ⋀ x ∉C)
⇔ x ∈ A - B ⋀ x ∈ A - C
⇔ x ∈ (A - B) ∩ (A - C)
Hence A - (B ∪ C) = (A − B) ∩ (A - C)
Proof for 12:
(a) A ∪ B = U ⋀ A ∩ B = ∅ ⇒ B = $\overline{A}$?
(b) Try to prove B ⊆ $\overline{A}$ and $\overline{A}$ ⊆ B.
Let x be an arbitrary element in the universe.
Then if x ∈ B, then x ∉A since A ∩ B = ∅. Hence x ∈ $\overline{A}$.
Hence B ⊆ $\overline{A}$.
If x ∈ $\overline{A}$, then x ∉A. Since x ∈ A ∨ x ∈ B (from A ∪ B = U), x ∈ B
must hold. Hence $\overline{A}$ ⊆ B.
Hence B = $\overline{A}$. (c) B = $\overline{A}$ ⇒ A ∪ B = U ⋀ A ∩ B = ∅ ?
Since B = $\overline{A}$, A ∪ B = A ∪ (U-A) = A ∪ U = U since A ⊆ U
Also A ∩ B = A ∩ (U - A) = ∅ by 10 above.
Proof for 13: Since $\overline{\overline{A}}$ ∪ $\overline{A}$ = $\overline{A}$ ∪ $\overline{\overline{A}}$, $\overline{A}$ ∪ $\overline{\overline{A}}$= U
Also since $\overline{\overline{A}}$ ∩ $\overline{A}$ = $\overline{A}$ ∩ $\overline{\overline{A}}$, $\overline{A}$ ∩ $\overline{\overline{A}}$ = ∅.
Hence $\overline{\overline{A}}$ satisfies the conditions for the complement of $\overline{A}$.
Hence $\overline{\overline{A}}$ = A.

5.1.4 Questions and Exercises

1. Determine whether each of the following pairs of sets is equal.
   - ∅ and {∅}
   - {a,b,c} and {a,b,c,c}
   - {1,2,3} and {{1},{2},{3}}
2. Let S1={a,b,c}, S2={a,b}, S3={b,c} and S4={b,c,d}. Which of the followings is correct?
   - S2 ⊆ S1
   - S3 ⊂S1
   - S4 ⊆ S1
3. Let A={a,b,c,d} and B={a,b,c,d,e,f}. Which of the followings is correct?
   - A ∪ B = {a,b,c,d,e,f}
   - A ∩ B = {a,b,c,d}
   - A − B = {e,f}
   - B − A = {e,f}
4. List the members of the following sets.

- $\{\mathbf{x} \mid \mathbf{x}$ is a real number such that $\mathbf{x}^2 = 4\}$
- $\{\mathbf{x} \mid \mathbf{x}$ is an integer such that $\mathbf{x}^2 = 2\}$

5. Determine whether each of the following pairs of sets is equal.

1. {1, 2, 1, 3, 1, 2}, {2, 3, 1}
2. {{1}}, {1, {1}}
3. $\emptyset$, {$\emptyset$}
4. For each of the following sets, determine whether 1 is an element of that set.

1. $\{\mathbf{x} \in \mathrm{R} \mid \mathbf{x}$ is an integer greater than 1$\}$
2. $\{\mathbf{x} \in \mathrm{R} \mid \mathbf{x}$ is the square of an integer$\}$
3. {1, 2, {1}}
4. {{1}, {{1}}}
5. {{1, 2}, {1, {1}}}
6. {{{1}}}
7. Suppose that **A**, **B**, and **C** are sets such that **A** $\subseteq$**B** and **B** $\subseteq$**C**. Show that **A** $\subseteq$**C**.
8. Find the power set of each of the following sets.

1. {a}
2. {a, {a}}
3. {$\emptyset$, {$\emptyset$}}
4. Let **A** be a set. Show that $\emptyset$x **A** = **A** x $\emptyset = \emptyset$.
5. How many different elements does **A** x **B** have if **A** has **m** elements and **B** has **n** elements?
6. Let **A** = {1, 2, 3, 4} and **b** = {0, 3, 5}. Find

1. **A** $\cup$**B**
2. **A** $\cap$**B**
3. **A - B**
4. **B - A**
5. What can you say about the sets **A** and **B** if the following are true?

1. **A** $\cup$**B**
2. **A** $\cup$**B** = **A**
3. **A - B** = **A**
4. Let **A** and **B** be subsets of a universal set **U**. Show that **A** $\subseteq$**B** if and only if $\overline{B} \subseteq \overline{A}$.
5. Let **A** and **B** be sets. Show that

1. **A** $\cup$**B** = **B** $\cup$**A.**
2. **A** $\cap$**B** = **B** $\cap$**A**.
3. Show that if **A** and **B** are sets, then $\overline{(A \cup B)} = \overline{A} \cap \overline{B}$ by showing each side is a subset of the other side.
4. Let **A**, **B**, and **C** be sets. Show that **A** $\cup$(**B** $\cup$**C**) = (**A** $\cup$**B**) $\cup$**C.**

Chapter 6

Discrete Structures Recursion[1]

6.1 Recursion

6.1.1 Recursive Definition

6.1.1.1 Recursive Definition

Sets which have too many elements to list them up, and for which there are no convenient or obvious predicates to specify their elements can often be defined using a recursive definition (also called inductive definition). It essentially gives a procedure to generate the members of the set one by one starting with some subset of its elements. In this type of definition, first a collection of elements to be included initially in the set is specified. These elements can be viewed as the seeds of the set being defined. Next, the rules to be used to generate elements of the set from elements already known to be in the set (initially the seeds) are given. These rules provide a method to construct the set element by element starting with the seeds. These rules can also be used to test elements for the membership in the set.

A recursive definition of a set always consists of three distinct clauses:

1. The basis clause (or simply basis) of the definition establishes that certain objects are in the set. This part of the definition specifies the "seeds" of the set from which the elements of the set are generated using the methods given in the inductive clause. The set of elements specified here is called basis of the set being defined.

2. The inductive clause (or simply induction) of the definition establishes the ways in which elements of the set can be combined to produce new elements of the set. The inductive clause always asserts that if objects are elements of the set, then they can be combined in certain specified ways to create other objects. Let us call the objects used to create a new object the parents of the new object, and the new object is their child.

3. The extremal clause asserts that unless an object can be shown to be a member of the set by applying the basis and inductive clauses a finite number of times, the object is not a member of the set.

The set you are trying to define recursively is the set that satisfies those three clauses.

There are a number of other ways of expressing the extremal clause that are equivalent to the extremal clause given above.

Examples of Recursive Definition of Set

Example 1. Definition of the Set of Natural Numbers N

The set N is the set that satisfies the following three clauses:

Basis Clause: $0 \in N$

Inductive Clause: For any element x in N, x + 1 is in N.

Extremal Clause: Nothing is in N unless it is obtained from the Basis and Inductive Clauses.

[1] This content is available online at <http://cnx.org/content/m15774/1.1/>.

The basis for this set N is { 0 } . The x + 1 in the Inductive Clause is the parent of x, and x is the child of x + 1. Following this definition, the set of natural numbers N can be obtained as follows:

First by the Basis Clause, 0 is put into N. Then by the Inductive Clause, since 0 is in N, 0 + 1 (= 1) is in N. 0 is the parent of 1, and 1 is the child of 0. Then by the Inductive Clause again, 1 + 1 (= 2) is in N. 1 is the parent of 2, and 2 is the child of 1. Proceeding in this manner all the "natural numbers" are put into N.

Note that if we don't have the Extremal Clause, 0.5, 1.5, 2.5, ... can be included in N, which is not what we want as the set of natural numbers.

Example 2. Definition of the Set of Nonnegative Even Numbers NE

The set NE is the set that satisfies the following three clauses:

Basis Clause: 0 ∈ NE

Inductive Clause: For any element x in NE, x + 2 is in NE.

Extremal Clause: Nothing is in NE unless it is obtained from the Basis and Inductive Clauses.

Example 3. Definition of the Set of Even Integers EI

The set EI is the set that satisfies the following three clauses:

Basis Clause: 0 ∈ EI

Inductive Clause: For any element x in EI, x + 2, and x - 2 are in EI.

Extremal Clause: Nothing is in EI unless it is obtained from the Basis and Inductive Clauses.

Example 4. Definition of the Set of Strings S over the alphabet {a,b} excepting empty string. This is the set of strings consisting of a's and b's such as abbab, bbabaa, etc.

The set S is the set that satisfies the following three clauses:

Basis Clause: a ∈ S, and b ∈ S.

Inductive Clause: For any element x in S, ax ∈ S, and bx ∈ S.

Here ax means the concatenation of a with x.

Extremal Clause: Nothing is in S unless it is obtained from the Basis and Inductive Clauses.

Tips for recursively defining a set:

For the "Basis Clause", try simplest elements in the set such as smallest numbers (0, or 1), simplest expressions, or shortest strings. Then see how other elements can be obtained from them, and generalize that generation process for the "Inductive Clause".

The set of propositions (propositional forms) can also be defined recursively.

6.1.1.2 Generalized Set Operations

As we saw earlier, union, intersection and Cartesian product of sets are associative. For example (A ∪ B) ∪ C = A ∪ (B ∪ C)

To denote either of these we often use A ∪ B ∪ C.

This can be generalized for the union of any finite number of sets as A1 ∪ A2 ∪.... ∪ An.

which we write as

$\bigcup_{i=1}^{n} A_i$

This generalized union of sets can be rigorously defined as follows:

Definition ($\bigcup_{i=1}^{n} A_i$):

Basis Clause: For n = 1, $\bigcup_{i=1}^{n} A_i = A_1$.

Inductive Clause: $\bigcup_{i=1}^{n+1} A_i = \bigcup_{i=1}^{n} A_i \cup$ An+1

Similarly the generalized intersection $\bigcap_{i=1}^{n} A_i$ and generalized Cartesian product $\prod_{i=1}^{n} A_i$ can be defined.

Based on these definitions, De Morgan's law on set union and intersection can also be generalized as follows:

Theorem (Generalized De Morgan)

$\overline{\bigcup_{i=1}^{n} A_i} = \bigcap_{i=1}^{n} \overline{A_i}$, and

$\overline{\bigcap_{i=1}^{n} A_i} = \bigcup_{i=1}^{n} \overline{A_i}$

Proof: These can be proven by induction on n and are left as an exercise.

6.1.1.3 Recursive Definition of Function

Some functions can also be defined recursively.

Condition: The domain of the function you wish to define recursively must be a set defined recursively.

How to define function recursively: First the values of the function for the basis elements of the domain are specified. Then the value of the function at an element, say x, of the domain is defined using its value at the parent(s) of the element x.

A few examples are given below.

They are all on functions from integer to integer except the last one.

Example 5: The function f(n) = n! for natural numbers n can be defined recursively as follows:

The function f is the function that satisfies the following two clauses:

Basis Clause: f(0) = 0! = 1

Inductive Clause: For all natural number n, f(n+1) = (n+1) f(n).

Note that here Extremal Clause is not necessary, because the set of natural numbers can be defined recursively and that has the extremal clause in it. So there is no chance of other elements to come into the function being defined.

Using this definition, 3! can be found as follows:

Since 0 ! = 1, 1 ! = 1 * 0 ! = 1 * 1 = 1 ,

Hence 2 ! = 2 * 1 ! = 2 * 1 = 2 .

Hence 3 ! = 3 * 2 ! = 3 * 2 * 1 = 6 .

Example 6: The function f(n) = 2n + 1 for natural numbers n can be defined recursively as follows:

The function f is the function that satisfies the following two clauses:

Basis Clause: f(0) = 1

Inductive Clause: For all natural number n, f(n+1) = f(n) + 2 .

See above for the extremal clause.

Example 7: The function f(n) = 2n for natural numbers n can be defined recursively as follows:

The function f is the function that satisfies the following two clauses:

Basis Clause: f(0) = 1

Inductive Clause: For all natural number n, f(n+1) = 2 f(n) .

See Example 5 for the extremal clause.

Example 8: The function L from the set S of strings over {a, b} to the set of natural numbers that gives the length of a string can be defined recursively as follows:

The function L is the function that satisfies the following two clauses:

Basis Clause: For symbols a and b of the alphabet, L(a) = 1 and L(b) = 1.

Inductive Clause: For any string x and y of S, L(xy) = L(x) + L(y) , where xy is the concatenation of strings x and y.

See Example 5 for the extremal clause.

This function L gives the number of a's and b's.

6.1.2 Recursive Algorithm

A recursive algorithm is an algorithm which calls itself with "smaller (or simpler)" input values, and which obtains the result for the current input by applying simple operations to the returned value for the smaller (or simpler) input. More generally if a problem can be solved utilizing solutions to smaller versions of the same problem, and the smaller versions reduce to easily solvable cases, then one can use a recursive algorithm to solve that problem. For example, the elements of a recursively defined set, or the value of a recursively defined function can be obtained by a recursive algorithm.

If a set or a function is defined recursively, then a recursive algorithm to compute its members or values mirrors the definition. Initial steps of the recursive algorithm correspond to the basis clause of the recursive definition and they identify the basis elements. They are then followed by steps corresponding to the inductive clause, which reduce the computation for an element of one generation to that of elements of the immediately preceding generation.

In general, recursive computer programs require more memory and computation compared with iterative algorithms, but they are simpler and for many cases a natural way of thinking about the problem.

Example 1: Algorithm for finding the k-th even natural number Note here that this can be solved very easily by simply outputting 2*(k - 1) for a given k . The purpose here, however, is to illustrate the basic idea of recursion rather than solving the problem.

Algorithm 1: Even(positive integer k)
Input: k , a positive integer
Output: k-th even natural number (the first even being 0)
Algorithm:
if k = 1, then return 0;
else return Even(k-1) + 2 .

Here the computation of Even(k) is reduced to that of Even for a smaller input value, that is Even(k-1). Even(k) eventually becomes Even(1) which is 0 by the first line. For example, to compute Even(3), Algorithm Even(k) is called with k = 2. In the computation of Even(2), Algorithm Even(k) is called with k = 1. Since Even(1) = 0, 0 is returned for the computation of Even(2), and Even(2) = Even(1) + 2 = 2 is obtained. This value 2 for Even(2) is now returned to the computation of Even(3), and Even(3) = Even(2) + 2 = 4 is obtained.

As can be seen by comparing this algorithm with the recursive definition of the set of nonnegative even numbers, the first line of the algorithm corresponds to the basis clause of the definition, and the second line corresponds to the inductive clause.

By way of comparison, let us see how the same problem can be solved by an iterative algorithm.

Algorithm 1-a: Even(positive integer k)
Input: k, a positive integer
Output: k-th even natural number (the first even being 0)
Algorithm:

```
int   i, even;
i := 1;
even := 0;
while( i < k ) {
        even := even + 2;
        i := i + 1;
}
return even .
```

Example 2: Algorithm for computing the k-th power of 2
Algorithm 2 Power_of_2(natural number k)
Input: k , a natural number
Output: k-th power of 2
Algorithm:
if k = 0, then return 1;
else return 2*Power_of_2(k - 1) .
By way of comparison, let us see how the same problem can be solved by an iterative algorithm.
Algorithm 2-a Power_of_2(natural number k)
Input: k , a natural number
Output: k-th power of 2
Algorithm:

```
int   i, power;
i := 0;
power := 1;
while( i < k ) {
        power := power * 2;
        i := i + 1;
```

}
return power .

The next example does not have any corresponding recursive definition. It shows a recursive way of solving a problem.

Example 3: Recursive Algorithm for Sequential Search
Algorithm 3 SeqSearch(L, i, j, x)
Input: L is an array, i and j are positive integers, i $\leq$j, and x is the key to be searched for in L.
Output: If x is in L between indexes i and j, then output its index, else output 0.
Algorithm:
if i $\leq$j , then
{
  if L(i) = x, then return i ;
  else return SeqSearch(L, i+1, j, x)
}
else return 0.

Recursive algorithms can also be used to test objects for membership in a set.

Example 4: Algorithm for testing whether or not a number x is a natural number
Algorithm 4 Natural(a number x)
Input: A number x
Output: "Yes" if x is a natural number, else "No"
Algorithm:
if x < 0, then return "No"
else
  if x = 0, then return "Yes"
  else return Natural(x - 1)

Example 5: Algorithm for testing whether or not an expression w is a proposition (propositional form)
Algorithm 5 Proposition(a string w)
Input: A string w
Output: "Yes" if w is a proposition, else "No"
Algorithm:
if w is 1(true), 0(false), or a propositional variable, then return "Yes"
else if w = $\sim$w1, then return Proposition(w1)
  else
    if (w = w1 $\vee$w2 or w1 $\bigwedge$w2 or w1 $\rightarrow$w2 or w1 $\leftrightarrow$w2) and
      Proposition(w1) = Yes and Proposition(w2) = Yes
    then return Yes
    else return No
end

6.1.3 Proof by Induction

6.1.3.1 Mathematical Induction – First Principle

As we have seen in recursion, the set of natural numbers can be defined recursively, and its elements can be generated one by one starting with 0 by adding 1. Thus the set of natural numbers can be described completely by specifying the basis element (0), and the process of generating an element from a known element in the set.

Taking advantage of this, natural numbers can be proven to have certain properties as follows:

First it is proven that the basis element, that is 0, has the property in question (basis step). You prove that the seeds (the first generation elements) have the property. Then it is proven that if an arbitrary natural number, denote it by n, has the property in question, then the next element, that is n + 1, has that

property (inductive step). Here you prove that the property is inherited from one generation (n) to the next generation (n + 1).

When these two are proven, then it follows that all the natural numbers have that property. For since 0 has the property by the basis step, the element next to it, which is 1, has the same property by the inductive step. Then since 1 has the property, the element next to it, which is 2, has the same property again by the inductive step. Proceeding likewise, any natural number can be shown to have the property. This process is somewhat analogous to the knocking over a row of dominos with knocking over the first domino corresponding to the basis step.

More generally mathematical statements involving a natural number n such as 1 + 2 + ... + n = n(n + 1)/2 can be proven by mathematical induction by the same token.

To prove that a statement P(n) is true for all natural number n≥n0, where n0 is a natural number, we proceed as follows:

Basis Step: Prove that P(n0) is true.

Induction: Prove that for any integer k≥n0, if P(k) is true (called induction hypothesis), then P(k+1) is true.

The first principle of mathematical induction states that if the basis step and the inductive step are proven, then P(n) is true for all natural number n≥n0.

As a first step for proof by induction, it is often a good idea to restate P(k+1) in terms of P(k) so that P(k), which is assumed to be true, can be used.

Example:

Prove that for any natural number n, 0 + 1 + ... + n = n(n + 1)/2 .

Proof:

Basis Step: If n = 0, then LHS = 0, and RHS = 0 * (0 + 1) = 0 .

Hence LHS = RHS.

Induction: Assume that for an arbitrary natural number n, 0 + 1 + ... + n = n(n + 1)/2 .

——— Induction Hypothesis

To prove this for n+1, first try to express LHS for n+1 in terms of LHS for n, and somehow use the induction hypothesis.

Here let us try

LHS for n + 1 = 0 + 1 + ... + n + (n + 1) = (0 + 1 + ... + n) + (n + 1).

Using the induction hypothesis, the last expression can be rewritten as

n(n + 1)/2 + (n + 1) .

Factoring (n + 1) out, we get

(n + 1)(n + 2) / 2,

which is equal to the RHS for n+1.

Thus LHS = RHS for n+1.

End of Proof.

6.1.3.2 Example of Use of Mathematical Induction — Program Correctness

Loops in an algorithm/program can be proven correct using mathematical induction. In general it involves something called "loop invariant" and it is very difficult to prove the correctness of a loop. Here we are going to give a few examples to convey the basic idea of correctness proof of loop algorithms.

First consider the following piece of code that computes the square of a natural number:

(We do not compute the square this way but this is just to illustrate the concept of loop invariant and its proof by induction.)

```
SQUARE Function: SQ(n)
S <- 0
i <- 0
while i < n
   S <- S + n
   i <- i + 1
```

return S

Let us first see how this code computes the square of a natural number. For example let us compute 3 2 using it.

First S <- 0 and i <- 0 give S = 0 and i = 0 initially.

Since i < 3, the while loop is entered.

S <- 0 + 3

i <- 0 + 1

producing S = 3 and i = 1.

Since i < 3, the while loop is entered the second time.

S <- 3 + 3

i <- 1 + 1

producing S = 6 and i = 2.

Since i < 3, the while loop is entered the third time.

S <- 6 + 3

i <- 2 + 1

producing S = 9 and i = 3.

Since i = 3, the while loop is not entered any longer, S = 9 is returned and the algorithm is terminated.

In general to compute n2 by this algorithm, n is added n times.

To prove that the algorithm is correct, let us first note that the algorithm stops after a finite number of steps. For i increases one by one from 0 and n is a natural number. Thus i eventually becomes equal to n.

Next, to prove that it computes n2, we show that after going through the loop k times, S = k*n and i = k hold. This statement is called a loop invariant and mathematical induction can be used to prove it.

Proof by induction.

Basis Step: k = 0. When k = 0, that is when the loop is not entered, S = 0 and i = 0. Hence S = k*n and i = k hold.

Induction Hypothesis: For an arbitrary value m of k, S = m * n and i = m hold after going through the loop m times.

Inductive Step: When the loop is entered (m + 1)-st time, S = m*n and i = m at the beginning of the loop. Inside the loop,

S <- m*n + n

i <- i + 1

producing S = (m + 1)*n and i = m + 1.

Thus S = k*n and i = k hold for any natural number k.

Now, when the algorithm stops, i = n. Hence the loop will have been entered n times.

Thus S = n*n = n2. Hence the algorithm is correct.

The next example is an algorithm to compute the factorial of a positive integer.

FACTORIAL Function: FAC(n)

i <- 1

F <- 1

while i < = n

F <- F * i

i <- i + 1

return F

Let us first see how this code computes the factorial of a positive integer. For example let us compute 3!.

First i <- 1 and F <- 1 give i = 1 and F = 1 initially.

Since i < 3, the while loop is entered.

F <- 1 * 1

i <- 1 + 1

producing F = 1 and i = 2.

Since i < 3, the while loop is entered the second time.

F <- 1 * 2

i <- 2 + 1
producing F = 2 and i = 3.
Since i = 3, the while loop is entered the third time.
F <- 2 * 3
i <- 3 + 1
producing F = 6 and i = 4.
Since i = 4, the while loop is not entered any longer, F = 6 is returned and the algorithm is terminated.

To prove that the algorithm is correct, let us first note that the algorithm stops after a finite number of steps. For i increases one by one from 1 and n is a positive integer. Thus i eventually becomes equal to n.

Next, to prove that it computes n!, we show that after going through the loop k times, F = k ! and i = k + 1 hold. This is a loop invariant and again we are going to use mathematical induction to prove it.

Proof by induction.

Basis Step: k = 1. When k = 1, that is when the loop is entered the first time, F = 1 * 1 = 1 and i = 1 + 1 = 2. Since 1! = 1, F = k! and i = k + 1 hold.

Induction Hypothesis: For an arbitrary value m of k, F = m! and i = m + 1 hold after going through the loop m times.

Inductive Step: When the loop is entered (m + 1)-st time, F = m! and i = (m+1) at the beginning of the loop. Inside the loop,
F <- m!* (m + 1)
i <- (m + 1) + 1
producing F = (m + 1)! and i = (m + 1) + 1.
Thus F = k! and i = k + 1 hold for any positive integer k.

Now, when the algorithm stops, i = n + 1. Hence the loop will have been entered n times. Thus F = n! is returned. Hence the algorithm is correct.

6.1.3.3 Mathematical Induction – Second Principle

There is another form of induction over the natural numbers based on the second principle of induction to prove assertions of the form ∀x P(x). This form of induction does not require the basis step, and in the inductive step P(n) is proved assuming P(k) holds for all k < n . Certain problems can be proven more easily by using the second principle than the first principle because P(k) for all k < n can be used rather than just P(n - 1) to prove P(n).

Formally the second principle of induction states that
if ∀n [∀k [k < n →P(k)] →P(n)] , then ∀n P(n) can be concluded.
Here ∀k [k < n →P(k)] is the induction hypothesis.

The reason that this principle holds is going to be explained later after a few examples of proof. Example 1: Let us prove the following equality using the second principle:
For any natural number n , 1 + 3 + ... + (2n + 1) = (n + 1)2.
Proof: Assume that 1 + 3 + ... + (2k + 1) = (k + 1)2 holds for all k, k < n.
Then 1 + 3 + ... + (2n + 1) = (1 + 3 + ... + (2n - 1)) + (2n + 1)
= n2 + (2n + 1) = (n + 1)2 by the induction hypothesis.
Hence by the second principle of induction 1 + 3 + ... + (2n + 1) = (n + 1)2 holds for all natural numbers.

Example 2: Prove that for all positive integer n, $\sum_{i=1}^{n}$i (i!) = (n + 1)! - 1
Proof: Assume that
1 * 1! + 2 * 2! + ... + k * k! = (k + 1)! - 1 for all k, k < n.
Then 1 * 1! + 2 * 2! + ... + (n - 1) * (n - 1)! + n * n!
= n! - 1 + n * n! by the induction hypothesis.
= (n + 1)n! - 1
Hence by the second principle of induction $\sum_{i=1}^{n}$i (i!) = (n + 1)! - 1 holds for all positive integers.

Example 3: Prove that any positive integer n, n > 1, can be written as the product of prime numbers.

Proof: Assume that for all positive integers k, n > k > 1, k can be written as the product of prime numbers.

We are going to prove that n can be written as the product of prime numbers.

Since n is an integer, it is either a prime number or not a prime number. If n is a prime number, then it is the product of 1, which is a prime number, and itself. Therefore the statement holds true.

If n is not a prime number, then it is a product of two positive integers, say p and q. Since both p and q are smaller than n, by the induction hypothesis they can be written as the product of prime numbers (Note that this is not possible, or at least very hard, if the First Principle is being used). Hence n can also be written as the product of prime numbers.

6.1.4 Questions and Exercises

1. Indicate which of the following statements are correct and which are not.

a. The number 23 can be generated for EI in Example 3 in Section Recursive Definition.

b. Basis and Inductive Clauses are sufficiency for membership for the set.

c. The set {4} can replace the basis for NE of Example 2 in Section Recursive Definition.

d. If empty set is the basis of S in Example 4 in Section Recursive Definition, then the string ab is in S.

2. Indicate which of the following statements are correct and which are not.

a. Algorithm 2 in Section Recursive Algorithm produces 0, 2, 6 and 8 when computing the third power of 2.

b. Recursive algorithms are good because they run more efficiently than iterative ones.

c. In Algorithm 3 in Section Recursive Algorithm, x is first compared with the key at the middle of L.

d. If the input to Algorithm 1 in Section Recursive Algorithm is not a natural number, then 0 is returned.

3. Look at the Section Mathematics Induction; indicate which of the following statements are correct and which are not.

a. In the Inductive Step, P(n) is proven assuming that P holds for the parent of n.

b. In the Inductive Step, since we assume P(k) for an arbitrary k, P(k+1) holds.

c. The Induction Hypothesis does NOT assume P(k) for all k.

d. In the Induction, since k is arbitrary, we can prove P(6) assuming P(5) holds.

e. The Basis Step proves the statement for the elements of the basis.

4. Look at the Section Mathematics Induction; indicate which of the following statements are correct and which are not.

a. In the Second Principle, P(k) is assumed true for one arbitrary value of k.

b. The Second Principle does not make a proof any easier.

c. The Basis Step of the First Principle is implicitly proven by the Second Principle.

d. The Second Principle can be applied when n starts at some integer larger than 0.

e. The Second Principle gives you more assumptions to use, making a proof easier.

5. Let **A**i = { 1, 2, 3, ..., i } for **i** = 1, 2, 3, ... Find $\bigcup_{i=1}^{n} A_i$

6. Let **A**i = { i, i+1, i+2, ... } for **i** = 1, 2, 3, ... Find $\text{intersect}_{i=1}^{n} A_i$

7. Give a recursive definition of the set of positive integers that are multiples of 5.

8. Give a recursive definition of

a. the set of even integers.

b. the set of positive integers congruent to 2 modulo 3.

c. the set of positive integers not divisible by 5.

9. When does a string belong to the set **A** of bit strings (i.e. strings of 0's and 1's) defined recursively by

Basis Clause: $\emptyset \in \mathbf{A}$

Inductive Clause: $0\mathbf{x}1 \in \mathbf{A}$ if $\mathbf{x} \in \mathbf{A}$

where $\emptyset$ is the empty string (An empty string is a string with no symbols in it.)

Extremal Clause: Nothing is in **A** unless it is obtained from the Basis and Inductive Clauses.

10. Find **f(1)**, **f(2)**, and **f(3)**, if **f(n)** is defined recursively by **f(0) = 2** and for **n = 0, 1, 2, ...**

a. f(n + 1) = f(n) + 2.

b. f(n + 1) = 3f(n).

c. f(n + 1) = 2f(n)**.**

11. Find **f(2)**, **f(3)**, and **f(4)**, if **f(n)** is defined recursively by **f(0) = 1**, **f(1) = -2** and for **n= 1, 2,...**

a. f(n + 1) = f(n) + 3f(n - 1).

b. f(n + 1) = f(n)2 f(n - 1).

12. Let **F** be the function such that **F(n)** is the sum of the first **n** positive integers. Give a recursive definition of **F(n)**.

13. Give a recursive algorithm for computing **nx** whenever **n** is a positive integer and **x** is an integer.

14. Give a recursive algorithm for finding the sum of the first **n** odd positive integers.

15. Use mathematical induction to prove that 3 + 3 * 5 + 3 * 52+ ... + 3 * 5n = 3 (5n+1 - 1)/4 whenever **n** is a nonnegative integer.

16. Prove that 12 + 32 + 52+ ... + (2**n** + 1)2 = (**n** + 1)(2**n** + 1)(2**n**+ 3)/3 whenever **n** is a nonnegative integer.

17. Show that 2n > **n**2whenever **n** is an integer greater than 4.

18. Show that any postage that is a positive integer number of cents greater than 7 cents can be formed using just 3-cent stamps and 5-cent stamps.

19. Use mathematical induction to show that 5 divides **n**5- **n** whenever **n** is a nonnegative integer.

20. Use mathematical induction to prove that if **A**1, **A**2, ...**A**n are subsets of a universal set **U**, then $\overline{\bigcap_{i=1}^{n} A_i} = \text{intersect}_{i=1}^{n} \overline{A}_i$

21. Find a formula for

1/2 + 1/4 + 1/8 + ... + 1/2n

by examining the values of this expression for small values of **n**. Use mathematical induction to prove your result.

22. Show that if **a**1, **a**2, ..., **a**n are **n** distinct real numbers, exactly **n - 1** multiplications are used to compute the product of these **n** numbers no matter how parentheses are inserted into their product. (**Hint**: Use the second principle of mathematical induction and consider the last multiplication).

Chapter 7

Discrete Structures Relation[1]

7.1 Relation

7.1.1 Introduction to Relation

The relation we are going to study here is an abstraction of relations we see in our everyday life such as those between parent and child, between car and owner, among name, social security number, address and telephone number etc. We are going to focus our attention on one key property which all the everyday relations have in common, define everything that has that property as a relation, and study properties of those relations. One of the places where relation in that sense is used is data base management systems. Along with hierarchical and network models of data, the relational model is widely used to represent data in a database. In this model the data in a database are represented as a collection of relations. Informally, each relation is like a table or a simple file. For example, consider the following table.

Employee		
Name	Address	Home Phone
Amy Angels	35 Mediterranean Av.	224-1357
Barbara Braves	221 Atlantic Av.	301-1734
Charles Cubs	312 Baltic Av.	223-9876

Table 7.1

Each row of this table represents a collection of data values such as name, address, and telephone number of a person. Each row is considered an instance of a relation and the table as the collection of the rows is considered a relation, which is the relation we are going to be studying in this chapter. Operations such as inserting or deleting entries to or from a table, merging two tables, finding the intersection of two tables, and searching for certain entries can be described simply and precisely as operations on relations, and known mathematical results on relations can be utilized without reinventing them. The relational model is flexible (easy to expand, easy to modify) and interface to query languages is simple. It is thus widely used today.

7.1.2 Definitions

7.1.2.1 Binary Relation

Here we are going to define relation formally, first binary relation, then general n-ary relation. A relation in everyday life shows an association of objects of a set with objects of other sets (or the same set) such

[1] This content is available online at <http://cnx.org/content/m15775/1.1/>.

as John owns a red Mustang, Jim has a green Miata etc. The essence of relation is these associations. A collection of these individual associations is a relation, such as the ownership relation between peoples and automobiles. To represent these individual associations, a set of "related" objects, such as John and a red Mustang, can be used. However, simple sets such as {John, a red Mustang} are not sufficient here. The order of the objects must also be taken into account, because John owns a red Mustang but the red Mustang does not own John, and simple sets do not deal with orders. Thus sets with an order on its members are needed to describe a relation. Here the concept of ordered pair and, more generally, that of ordered n-tuple are going to be defined first. A relation is then defined as a set of ordered pairs or ordered n-tuples.

Definition (ordered pair):

An ordered pair is a set of a pair of objects with an order associated with them. If objects are represented by x and y, then we write an ordered pair as <x, y> or <y, x>. In general <x, y> is different from <y, x>.

Definition (equality of ordered pairs):

Two ordered pairs <a, b> and <c, d> are equal if and only if a = c and b = d. For example, if the ordered pair <a, b> is equal to <1, 2>, then a = 1, and b = 2. <1, 2> is not equal to the ordered pair <2, 1>.

Definition (binary relation):

A binary relation from a set A to a set B is a set of ordered pairs <a, b> where a is an element of A and b is an element of B.

When an ordered pair <a, b> is in a relation R, we write a R b, or <a, b> ∈R. It means that element a is related to element b in relation R. When A = B, we call a relation from A to B a (binary) relation on A.

Definition (Cartesian product):

The set of all ordered pairs <a, b>, where a is an element of A and b is an element of B, is called the Cartesian product of A and B and is denoted by A × B.

Thus a binary relation from A to B is a subset of Cartesian product A × B.

Examples:

If A = {1, 2, 3} and B = {4, 5}, then {<1, 4>, <2, 5>, <3, 5>}, for example, is a binary relation from A to B.

However, {<1, 1>, <1, 4>, <3, 5>} is not a binary relation from A to B because 1 is not in B.

The parent-child relation is a binary relation on the set of people. <John, John Jr.>, for example, is an element of the parent-child relation if John is the father of John Jr.

7.1.2.2 Definition of n-ary Relation

Here we are going to formally define general n-ary relation using the concept of ordered n-tuple.

Definition (ordered n-tuple): An ordered n-tuple is a set of n objects with an order associated with them. If n objects are represented by x1, x2, ..., xn, then we write the ordered n-tuple as <x1, x2, ..., xn> .

Definition (Cartesian product): Let A1, ..., An be n sets. Then the set of all ordered n-tuples <x1, ..., xn> , where xi ∈Ai for all i, 1 ≤i ≤n , is called the Cartesian product of A1, ..., An, and is denoted by A1 ×... ×An .

Definition (equality of n-tuples): Two ordered n-tuples <x1, ..., xn> and <y1, ..., yn> are equal if and only if xi = yi for all i, 1 ≤i ≤n.

For example the ordered 3-tuple <1, 2, 3> can be equal to only <1, 2, 3> and nothing else. It is not equal to the ordered n-tuple <2, 3, 1> for example.

Definition (n-ary relation): An n-ary relation on sets A1, ..., An is a set of ordered n-tuples <a1, ..., an> where ai is an element of Ai for all i, 1 ≤i ≤n . Thus an n-ary relation on sets A1, ..., An is a subset of Cartesian product A1 ×... ×An .

Example 1: Let A1 be a set of names, A2 a set of addresses, and A3 a set of telephone numbers. Then a set of 3-tuples < name, address, telephone number > such as { < Amy Angels, 35 Mediterranean Ave, 224-1357 >, < Barbara Braves, 221 Atlantic Ave, 301-1734 >, < Charles Cubs, 312 Baltic Ave, 223-9876 > }, is a 3-ary (ternary) relation over A1, A2 and A3.

Example 2: Let A1 be a set of names. Then a set of 1-tuples such as {< Amy >, < Barbara >, < Charles >}, is a 1-ary (unary) relation over A1.

A unary relation represents a property/characteristic, such as tall, rich etc., shared by the members of A1 listed in the relation.

7.1.2.2.1 Special Relations

The empty set is certainly a set of ordered n-tuples. Therefore it is a relation. It is called the empty relation.

The Cartesian product A1 ×... ×An of sets A1 , ... , An , is also a relation, and it is called the universal relation.

7.1.2.3 Equality of Relations

7.1.2.3.1 Definition (equality of binary relation):

Two binary relations R1 ⊆A1 ×A2 and R2⊆B1 ×B2 are equal if and only if A1 = B1, A2 = B2 , and R1 = R2 as a set.

For example, let R1 = {<1, 2> , <2, 2>} ⊆{1, 2} ×{1, 2} , and R2 = {<a, b>, <b, b>} ⊆{a, b} ×{a, b} . Then R1 = R2 if and only if a = 1 and b = 2.

7.1.2.3.2 Definition (equality of n-ary relation):

An n-ary relation R1 ⊆A1 ×... ×An and an m-ary relation R2 ⊆B1 ×... ×Bm are equal if and only if m = n, Ai = Bi for each i, 1 ≤i ≤n , and R1 = R2 as a set of ordered n-tuples.

7.1.2.4 Recursive Definition of Relation

Certain relations can be defined recursively. Note that a relation is a set. Therefore a recursive definition of a relation follows the same format as that of sets. Here only examples are given.

Example 1: Let us define recursively the relation "less than" denoted by R< on the set of natural numbers N.

Note that R< = { <a, b> | a ∈N ⋀ b ∈N ⋀ a < b } = { <0, 1>, <0, 2> , ..., <1, 2>, ... }.

Basis Clause: <0, 1> ∈R< (or 0 R< 1), meaning 0 is less than 1.

Inductive Clause: For all x and y in N, if <x, y> ∈R< , then <x, y + 1> ∈R< , and <x + 1, y + 1> ∈R< .

Extremal Clause: Nothing is in R< , unless it is obtained from the Basis and Inductive Clauses.

Informally one can see that this definition is correct as follows:

First, <0, 1> is the "simplest" element in R< .

Next by arranging the ordered pairs in R< as follows, one can see that the two operations in the Inductive Clause generate them all:

<0, 1>, <0, 2>, <0, 3>, ... , <0, n>, ...
<1, 2>, <1, 3>, <1, 4>, ... , <1, n>, ...
<2, 3>, <2, 4>, <2, 5>, ... , <2, n>, ...
.....
.....

In each row, the first (leftmost) ordered pair gives the "smallest" ordered pair with the first number. For example <2, 3> is the smallest with 2 as the first component. It is obtained from the first ordered pair of the row immediately above it by using <x + 1, y + 1> of the Inductive Clause, apply that to <1, 2> to get <2, 3> , for example.

Within each row, the ordered pairs are obtained from the first one by using <x, y + 1> of the Inductive Clause successively.

Thus all the ordered pairs of R< are generated from <0, 1> by following the Inductive Clause.

Example 2: Let Ra + b = c be the set of triples of natural numbers <a, b, c> which satisfy a + b = c . Then Ra + b = c on the set of natural numbers N can be defined recursively as follows.

Basis Clause: <0, 0, 0> ∈Ra + b = c.

Inductive Clause: For all x, y and z in N, if <x, y, z> ∈Ra + b = c , then <x + 1, y, z + 1> and <x, y + 1, z + 1> ∈Ra + b = c.

Extremal Clause: Nothing is in Ra + b = c unless it is obtained from the Basis and Inductive Clauses.

7.1.3 Properties of Binary Relation, and Operations

7.1.3.1 Digraph

A digraph is short for directed graph, and it is a diagram composed of points called vertices (nodes) and arrows called arcs going from a vertex to a vertex.

For example Figure 1 is a digraph with 3 vertices and 4 arcs.

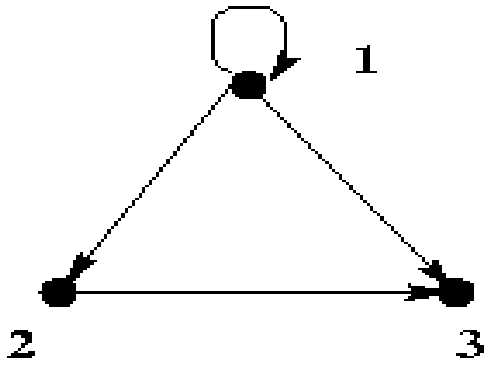


Figure 7.1

In this figure the vertices are labeled with numbers 1, 2, and 3.

Mathematically a digraph is defined as follows.

Definition (digraph): A digraph is an ordered pair of sets G = (V, A), where V is a set of vertices and A is a set of ordered pairs (called arcs) of vertices of V.

In the example, G1 , given above, V = { 1, 2, 3 } , and A = { <1, 1>, <1, 2>, <1, 3>, <2, 3> } .

7.1.3.2 Digraph representation of binary relations

A binary relation on a set can be represented by a digraph.

Let R be a binary relation on a set A, that is R is a subset of A×A. Then the digraph, call it G, representing R can be constructed as follows:

1. The vertices of the digraph G are the elements of A, and

2. <x, y> is an arc of G from vertex x to vertex y if and only if <x, y> is in R.

Example: The less than relation R on the set of integers A = {1, 2, 3, 4} is the set {<1, 2>, <1, 3>, <1, 4>, <2, 3> , <2, 4> , <3, 4> } and it can be represented by the digraph in Figure 2.

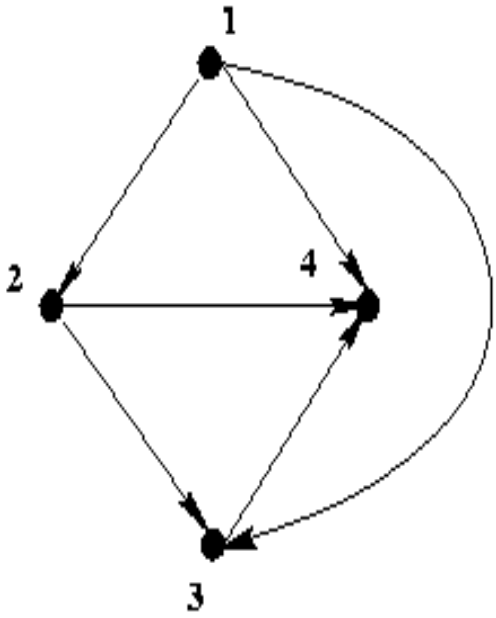


Figure 7.2

Let us now define some of the basic concepts on digraphs.

Definition (loop): An arc from a vertex to itself such as <1, 1>, is called a loop (or self-loop)

Definition (degree of vertex): The in-degree of a vertex is the number of arcs coming to the vertex, and the out-degree is the number of arcs going out of the vertex.

For example, the in-degree of vertex 2 in the digraph G2 shown above is 1, and the out-degree is 2.

Definition (path): A path from a vertex x0 to a vertex xn in a digraph G = (V, A) is a sequence of vertices x0, x1, ..., xn that satisfies the following:

for each i, $0 \leq i \leq n - 1$, $\langle xi , xi + 1\rangle \in A$, or $\langle xi + 1 , xi\rangle \in A$, that is, between any pair of vertices there is an arc connecting them. x0 is the initial vertex and xn is the terminal vertex of the path.

A path is called a directed path if $\langle xi, xi + 1\rangle \in A$, for every i, $0 \leq i \leq n - 1$.

If the initial and the terminal vertices of a path are the same, that is, $x0 = xn$, then the path is called a cycle.

If no arcs appear more than once in a path, the path is called a simple path. A path is called elementary if no vertices appear more than once in it except for the initial and terminal vertices of a cycle. In a simple cycle one vertex appears twice in the sequence: once as the initial vertex and once as the terminal vertex.

Note: There are two different definitions for "simple path". Here we follow the definition of Berge[1], Liu[2], Rosen[3] and others. A "simple path" according to another group (Cormen et al[4], Stanat and McAllister[5] and others) is a path in which no vertices appear more than once.

Definition(connected graph): A digraph is said to be connected if there is a path between every pair of its vertices.

Example: In the digraph G3 given in Figure 3,

1, 2, 5 is a simple and elementary path but not directed,

1, 2, 2, 5 is a simple path but neither directed nor elementary.

1, 2, 4, 5 is a simple elementary directed path,
1, 2, 4, 5, 2, 4, 5 is a directed path but not simple (hence not elementary),
1, 3, 5, 2, 1 is a simple elementary cycle but not directed, and
2, 4, 5, 2 is a simple elementary directed cycle.
This digraph is connected.

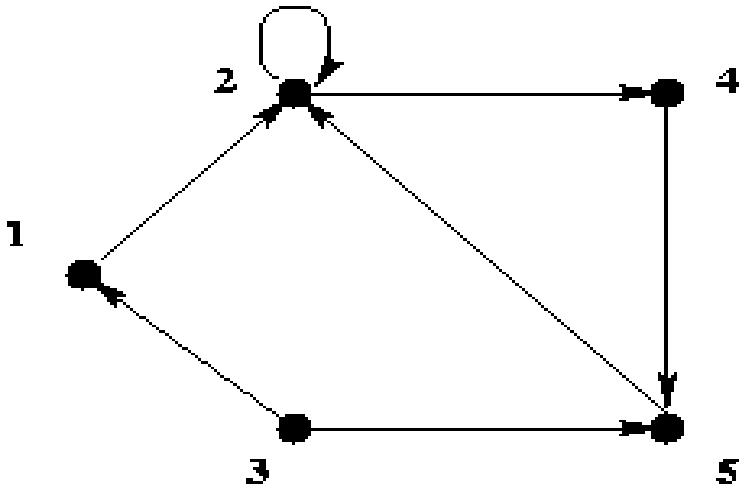


Figure 7.3

Sometimes we need to refer to part of a given digraph. A partial digraph of a digraph is a digraph consisting of arbitrary numbers of vertices and arcs of the given digraph, while a subdigraph is a digraph consisting of an arbitrary number of vertices and all the arcs between them of the given digraph. Formally they are defined as follows: Definition (subdigraph, partial digraph): Let G = (V, A) be a digraph. Then a digraph (V', A') is a partial digraph of G , if V' $\subseteq$V , and A' $\subseteq$A $\cap$(V' $\times$V') . It is a subdigraph of G , if V' $\subseteq$V , and A' = A $\cap$ (V'$\times$V')
A partial digraph and a subdigraph of G3 given above are shown in Figure 4.

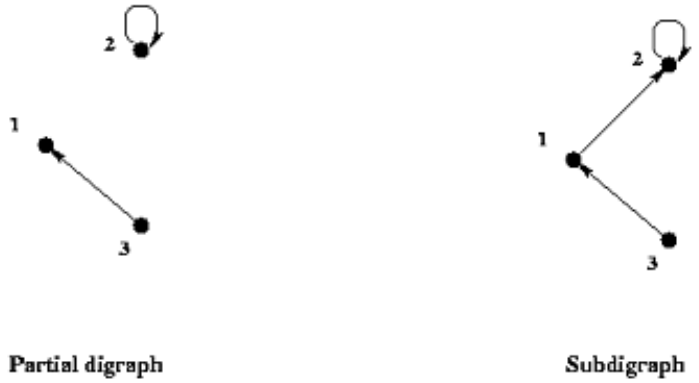


Figure 7.4

7.1.3.3 Properties of Binary Relation

Certain important types of binary relation can be characterized by properties they have. Here we are going to learn some of those properties binary relations may have. The relations we are interested in here are binary relations on a set.

Definition(reflexive relation): A relation R on a set A is called reflexive if and only if $< a, a > \in R$ for every element a of A.

Example 1: The relation $\leq$ on the set of integers {1, 2, 3} is {<1, 1>, <1, 2>, <1, 3>, <2, 2>, <2, 3>, <3, 3>} and it is reflexive because <1, 1>, <2, 2>, <3, 3> are in this relation. As a matter of fact $\leq$ on any set of numbers is also reflexive. Similarly $\geq$ and = on any set of numbers are reflexive. However, < (or >) on any set of numbers is not reflexive.

Example 2: The relation $\subseteq$on the set of subsets of {1, 2} is $\{ < \emptyset, \emptyset > , < \emptyset, \{1\} > , < \emptyset, \{2\} > , < \emptyset, \{1, 2\} > , < \{1\} , \{1\} > , < \{1\} , \{1, 2\} > , < \{2\} , \{2\} > , < \{2\} , \{1, 2\} > , < \{1, 2\} , \{1, 2\} > \}$ and it is reflexive. In fact relation $\subseteq$ on any collection of sets is reflexive.

Definition(irreflexive relation): A relation R on a set A is called irreflexive if and only if $<a, a> \notin R$ for every element a of A.

Example 3: The relation > (or <) on the set of integers {1, 2, 3} is irreflexive. In fact it is irreflexive for any set of numbers.

Example 4: The relation $\{< 1, 1 >, < 1, 2 >, < 1, 3 >, < 2, 3>, < 3, 3 >\}$ on the set of integers {1, 2, 3} is neither reflexive nor irreflexive.

Definition(symmetric relation): A relation R on a set A is called symmetric if and only if for any a, and b in A, whenever $<a, b> \in R$, $<b, a> \in R$.

Example 5: The relation = on the set of integers {1, 2, 3} is {<1, 1> , <2, 2> <3, 3> } and it is symmetric. Similarly = on any set of numbers is symmetric. However, < (or >), $\leq$ (or $\geq$ on any set of numbers is not symmetric.

Example 6: The relation "being acquainted with" on a set of people is symmetric.

Definition (antisymmetric relation): A relation R on a set A is called antisymmetric if and only if for any a, and b in A, whenever $<a, b> \in R$, and $<b, a> \in R$, a = b must hold. Equivalently, R is antisymmetric if and only if whenever $<a, b> \in R$, and $a \neq b$, $<b, a> \notin R$. Thus in an antisymmetric relation no pair of elements are related to each other.

Example 7: The relation < (or >) on any set of numbers is antisymmetric. So is the equality relation on any set of numbers.

Definition (transitive relation): A relation R on a set A is called transitive if and only if for any a, b, and c in A, whenever <a, b> ∈R , and <b, c> ∈R , <a, c> ∈R .

Example 8: The relation ≤ on the set of integers {1, 2, 3} is transitive, because for <1, 2> and <2, 3> in ≤, <1, 3> is also in ≤, for <1, 1> and <1, 2> in ≤, <1, 2> is also in ≤, and similarly for the others. As a matter of fact ≤ on any set of numbers is also transitive. Similarly ≥ and = on any set of numbers are transitive.

Figure 5 show the digraph of relations with different properties.

(a) is reflexive, antisymmetric, symmetric and transitive, but not irreflexive.

(b) is neither reflexive nor irreflexive, and it is antisymmetric, symmetric and transitive.

(c) is irreflexive but has none of the other four properties.

(d) is irreflexive, and symmetric, but none of the other three.

(e) is irreflexive, antisymmetric and transitive but neither reflexive nor symmetric.

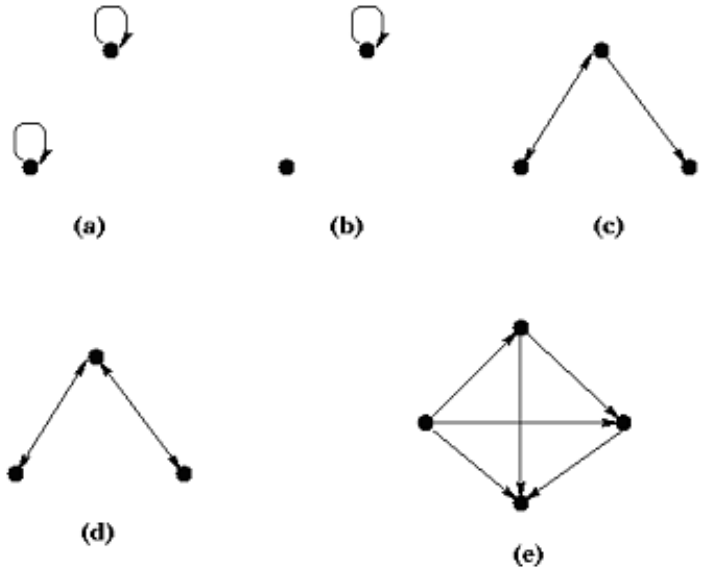


Figure 7.5

7.1.3.4 Operations on Binary Relations

7.1.3.4.1 Set Operations

A relation is a set. It is a set of ordered pairs if it is a binary relation, and it is a set of ordered n-tuples if it is an n-ary relation. Thus all the set operations apply to relations such as ∪, ∩, and complementing.

For example, the union of the "less than" and "equality" relations on the set of integers is the "less than or equal to" relation on the set of integers. The intersection of the "less than" and "less than or equal to" relations on the set of integers is the "less than" relation on the same set. The complement of the "less than" relation on the set of integers is the "greater than or equal to" relation on the same set.

7.1.3.4.2 Composite Relations

If the elements of a set A are related to those of a set B, and those of B are in turn related to the elements of a set C, then one can expect a relation between A and C. For example, if Tom is my father (parent-child relation) and Sarah is a sister of Tom (sister relation), then Sarah is my aunt (aunt-nephew/niece relation). Composite relations give that kind of relations.

Definition(composite relation): Let R1 be a binary relation from a set A to a set B, R2 a binary relation from B to a set C. Then the composite relation from A to C denoted by R1R2 (also denoted by R1 ∘ R2 is defined as

R1R2 = {<a, c> | a ∈A ⋀c ∈C ⋀∃b [b ∈B ⋀<a, b> ∈R1 ⋀<b, c> ∈R2] } .

In English, this means that an element a in A is related to an element c in C if there is an element b in B such that a is related to b by R1 and b is related to c by R2. Thus R1R2 is a relation from A to C via B in a sense. If R1 is a parent-child relation and R2 is a sister relation, then R1R2 is an aunt-nephew/niece relation.

Example 1: Let A = {a1 , a2} , B = {b1 , b2 , b3} , and C = {c1 , c2} . Also let R1 = {<a1 , b1> , <a1 , b2> , <a2 , b3> } , and R2 = {<b1 , c1> , <b2 , c1> , <b2 , c2> , <b3 , c1> } . Then R1R2 = {<a1 , c1> , <a1 , c2> , <a2 , c1> } .

This is illustrated in Figure 6. The dashed lines in the figure of R1R2 indicate the ordered pairs in R1R2, and dotted lines show ordered pairs that produce the dashed lines. (The lines in the left figure are all supposed to be solid lines.)

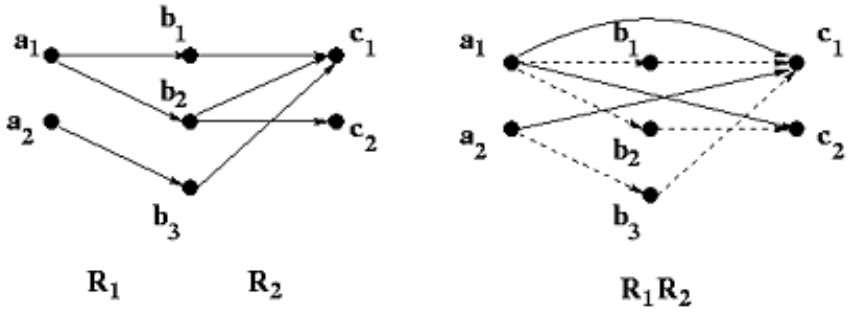


Figure 7.6

Example 2: If R is the parent-child relation on a set of people A, then RR, also denoted by R2, is the grandparent-grandchild relation on A.

More examples:

The digraphs of R2 for several simple relations R are shown in Figure 7:

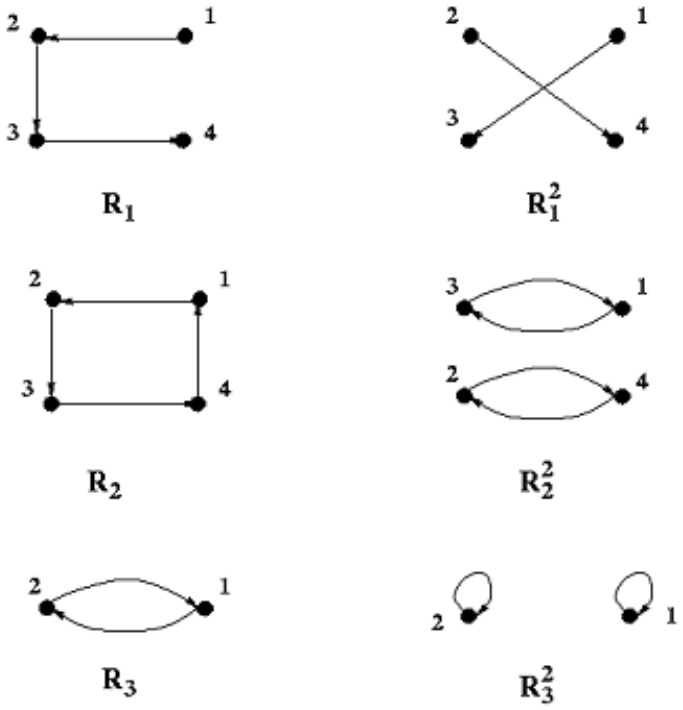


Figure 7.7

7.1.3.4.3 Properties of Composite Relations

Composite relations defined above have the following properties. Let R1 be a relation from A to B, and R2 and R3 be relations from B to C. Then

1. R1(R2R3) = (R1R2)R3
2. R1(R2 ∪R3) = R1R2 ∪R1R3
3. R1(R2 ∩R3) ⊆R1R2 ∩R1R3 Proofs for these properties are omitted.

7.1.3.4.4 Powers of Relation

Let R be a binary relation on A. Then Rn for all positive integers n is defined recursively as follows:

Definition (power of relation):

Basis Clause: R0 = E, where E is the equality relation on A.

Inductive Clause: For an arbitrary natural number n, Rn+1 = RnR.

Note that there is no need for extremal clause here.

Thus for example R1 = R, R2 = RR, and R3 = R2R = (RR)R = R(RR) = RRR.

The powers of binary relation R on a set A defined above have the following properties.

1. Rm+n = RmRn,
2. (Rm)n = Rmn.

7.1.4 Special Relations

7.1.4.1 Closure of Binary Relation

In our everyday life we often talk about parent-child relationship. This is a binary relation on the set of people in the world, dead or alive. Also we are often interested in ancestor-descendant relations. Although the latter relation can be obtained from the former, hence it is redundant in that sense, we do use ancestor-descendant relations which give us necessary information more directly. This ancestor-descendant relation relates two people if there is a sequence of parent-child relations from one to the other. It includes the parent-child relation as a subset. The ancestor-descendant relation is an example of the closure of a relation, in particular the transitive closure of the parent-child relation. In general, the closure of a relation is the smallest extension of the relation that has a certain specific property such as the reflexivity, symmetry or transitivity. Formally they are defined as follows:

Definition (reflexive closure): A relation R' is the reflexive closure of a relation R if and only if

(1) R' is reflexive,

(2) R $\subseteq$R', and

(3) for any relation R", if R $\subseteq$R" and R" is reflexive, then R' $\subseteq$R" , that is, R' is the smallest relation that satisfies (1) and (2).

Example: Let R be the less-than relation on the set of integers I, that is R = { < a, b> | a $\in$I $\bigwedge$b $\in$I $\bigwedge$a < b }.

Then the reflexive closure r(R) of R is the union of R and the equality relation on I, that is r(R) = {<a, b> | a $\in$I $\bigwedge$b $\in$I $\bigwedge$a $\leq$b }

The digraph of the reflexive closure of a relation is obtained from the digraph of the relation by adding a self-loop at each vertex if one is already not there.

Symmetric and transitive closures can be defined similarly.

Definition (symmetric closure): A relation R' is the symmetric closure of a relation R if and only if

(1) R' is symmetric,

(2) R $\subseteq$R', and

(3) for any relation R", if R $\subseteq$R", and R" is symmetric, then R' $\subseteq$R" , that is, R' is the smallest relation that satisfies (1) and (2).

Example: Let R be the less-than relation on the set of integers I. Then the symmetric closure of R, denoted by s(R) is s(R) = { <a, b> | a $\in$I $\bigwedge$ b $\in$I $\bigwedge$[a < b $\vee$ a > b] } that is { <a, b> | a $\in$I $\bigwedge$b $\in$I $\bigwedge$a $\neq$b }

The digraph of the symmetric closure of a relation is obtained from the digraph of the relation by adding for each arc the arc in the reverse direction if one is already not there.

Definition (transitive closure): A relation R' is the transitive closure of a relation R if and only if

(1) R' is transitive,

(2) R $\subseteq$R', and

(3) for any relation R", if R $\subseteq$R" and R" is transitive, then R' $\subseteq$R" , that is, R' is the smallest relation that satisfies (1) and (2).

Examples: The transitive closure of a parent-child relation is the ancestor-descendant relation as mentioned above, and that of the less-than relation on I is the less-than relation itself.

The digraph of the transitive closure of a relation is obtained from the digraph of the relation by adding for each directed path the arc that shunts the path if one is already not there.

Two more examples of closures are given in Figure 8 in terms of digraphs.

The arrows with two heads represent arrows going in opposite directions.

7.1.4.1.1 Properties of Closure

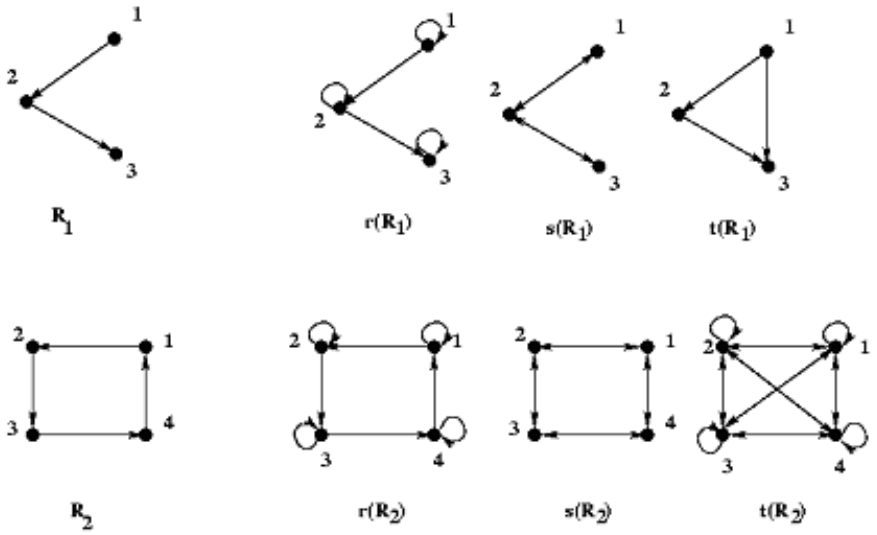


Figure 7.8

The closures have the following properties. They are stated here as theorems without proof.

Theorem: Let E denote the equality relation, and Rc the inverse relation of binary relation R, all on a set A, where Rc = {< a, b > | < b, a > ∈R}. Then

1. r(R) = R ∪ E
2. s(R) = R ∪Rc
3. t(R) = $\overset{\infty}{i=1}$Ri = $\overset{n}{i=1}$ Ri, if |A| = n.
4. R is reflexive if and only if r(R) = R.
5. R is symmetric if and only if s(R) = R.
6. R is transitive if and only if t(R) = R.

7.1.4.2 Equivalence Relation

On the face of most clocks, hours are represented by integers between 1 and 12. However, since a day has 24 hours after 12 hours, a clock goes back to hour 1, and starts all over again from there. Thus each pair of hours such as 1 and 13, 2 and 14, etc. share one number 1, 2, ...etc., respectively. The same applies when we are interested in more than 24 hours. 25th hour is 1, so are 37th, 49th etc. What we are doing here essentially is that we consider the numbers in each group such as 1, 13, 25, ..., equivalent in the sense that they all are represented by one number (they are congruent modulo 12). Being representable by one number such as we see on clocks is a binary relation on the set of natural numbers and it is an example of equivalence relation we are going to study here.

The concept of equivalence relation is characterized by three properties as follows:

Definition (equivalence relation): A binary relation R on a set A is an equivalence relation if and only if

(1) R is reflexive

(2) R is symmetric, and

(3) R is transitive.

Example 1: The equality relation (=) on a set of numbers such as {1, 2, 3} is an equivalence relation.

Example 2: The congruent modulo m relation on the set of integers i.e. {<a, b>| a ≡ b (mod m)}, where m is a positive integer greater than 1, is an equivalence relation.

Note that the equivalence relation on hours on a clock is the congruent mod 12, and that when m = 2, i.e. the congruent mod 2, all even numbers are equivalent and all odd numbers are equivalent. Thus the set of integers are divided into two subsets: evens and odds.

Example 3: Taking this discrete structures course together this semester is another equivalence relation.

Equivalence relations can also be represented by a digraph since they are a binary relation on a set. For example the digraph of the equivalence relation congruent mod 3 on {0, 1, 2, 3, 4, 5 , 6} is as shown in Figure 9. It consists of three connected components.

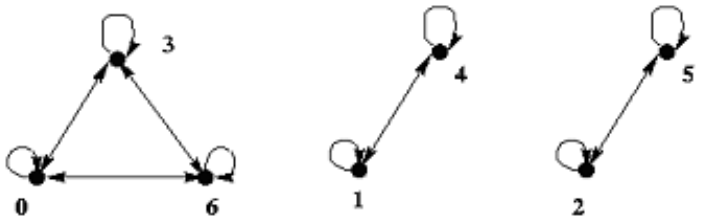


Figure 7.9

The set of even numbers and that of odd numbers in the equivalence relation of congruent mod 2, and the set of integers equivalent to a number between 1 and 12 in the equivalence relation on hours in the clock example are called an equivalence class. Formally it is defined as follows:

Definition (equivalence class): For an equivalence relation R on a set A, the set of the elements of A that are related to an element, say a, of A is called the equivalence class of element a and it is denoted by [a].

Example 4: For the equivalence relation of hours on a clock, equivalence classes are

[1] = {1, 13, 25, ... } = {1+ 12n: n ∈N} ,

[2] = {2, 14, 26, ... } = {2+ 12n: n ∈N} ,

.....,

where N is the set of natural numbers. There are altogether twelve of them.

For an equivalence relation R on a set A, every element of A is in an equivalence class. For if an element, say b, does not belong to the equivalence class of any other element in A, then the set consisting of the element b itself is an equivalence class. Thus the set A is in a sense **covered** by the equivalence classes. Another property of equivalence class is that equivalence classes of two elements of a set A are either disjoint or identical, that is either [a] = [b] or [a] ∩ [b] = ∅ for arbitrary elements a and b of A. Thus the set A is partitioned into equivalence classes by an equivalence relation on A. This is formally stated as a theorem below after the definition of partition.

Definition (partition): Let A be a set and let A1, A2, ..., An be subsets of A. Then {A1, A2, ..., An} is a partition of A, if and only if

(1) $\bigcup_{i=1}^{n}$Ai = A, and

(2) Ai ∩Aj = ∅, if Ai ≠ Aj , 1 ≤ i, j ≤ n .

(3) Example 5: Let A = {1, 2, 3, 4, 5}, A1 = {1, 5}, A2 = {3}, and A3 = {2, 4}. Then {A1, A2, A3} is a partition of A. However, B1 = {1, 2, 5}, B2 = {2, 3}, and B3 = {4} do not form a partition for A because B1 ∩B2 ≠∅, though B1 ≠B2.

Theorem 1: The set of equivalence classes of an equivalence relation on a set A is a partition of A.

Conversely, a partition of a set A determines an equivalence relation on A.

Theorem 2: Let {A1, ..., An} be a partition of a set A. Define a binary relation R on A as follows: <a, b> ∈R if and only if a ∈Ai and b ∈Ai for some i, 1 ≤i ≤n . Then R is an equivalence relation.

Theorem 3: Let R1 and R2 be equivalence relations. Then R1 $\cap$R2 is an equivalence relation, but R1 $\cup$R2 is not necessarily an equivalence relation.

7.1.4.3 Order Relation

Shoppers in a grocery store are served at a cashier on the first-come-first-served basis. When there are many people at cashiers, lines are formed. People in these lines are ordered for service: Those at the head of a line are served sooner than those at the end. Cars waiting for the signal to change at an intersection are also ordered similarly. Natural numbers can also be ordered in the increasing order of their magnitude. Those are just a few examples of order we encounter in our daily lives. The order relations we are going to study here are an abstraction of those relations. The properties common to orders we see in our daily lives have been extracted and are used to characterize the concepts of order. Here we are going to learn three types of order: partial order, total order, and quasi order.

Definition(partial order): A binary relation R on a set A is a partial order if and only if it is

(1) reflexive,

(2) antisymmetric, and

(3) transitive.

The ordered pair <A, R> is called a poset (partially ordered set) when R is a partial order.

Example 1: The less-than-or-equal-to relation on the set of integers I is a partial order, and the set I with this relation is a poset.

Example 2: The subset relation on the power set of a set, say {1, 2}, is also a partial order, and the set {1, 2} with the subset relation is a poset.

Definition(total order): A binary relation R on a set A is a total order if and only if it is

(1) a partial order, and

(2) for any pair of elements a and b of A, $< a, b > \in$R or $< b, a > \in$R.

That is, every element is related with every element one way or the other. A total order is also called a linear order.

Example 3: The less-than-or-equal-to relation on the set of integers I is a total order.

The strictly-less-than and proper-subset relations are not partial order because they are not reflexive. They are examples of some relation called quasi order.

Definition(quasi order): A binary relation R on a set A is a quasi order if and only if it is

(1) irreflexive, and

(2) transitive.

A quasi order is necessarily antisymmetric as one can easily verify.

Caution Like many other definitions there is another fairly widely used definition of quasi order in the literature. According to that definition a quasi order is a relation that is reflexive and transitive. That is something quite different from "quasi order" we have defined here.

Example 4: The less-than relation on the set of integers I is a quasi order.

Example 5: The proper subset relation on the power set of a set, say {1, 2}, is also a quasi order.

The concept of least/greatest number in a set of integers can be generalized for a general poset. We start with the concepts of minimal/maximal elements.

Definition(minimal/maximal element): Let $< A, \precsim >$ be a poset, where $\precsim$ represents an arbitrary partial order. Then an element b $\in$A is a minimal element of A if there is no element a $\in$A that satisfies a $\precsim$b. Similarly an element b $\in$A is a maximal element of A if there is no element a $\in$A that satisfies b $\precsim$a.

Example 6: The set of {{1}, {2}, {1, 2}} with $\subseteq$ has two minimal elements {1} and {2}. Note that {1}, and {2} are not related to each other in $\subseteq$. Hence we can not say which is "smaller than" which, that is, they are not comparable.

Definition(least/greatest element): Let $< A, \precsim >$ be a poset. Then an element b$\in$A is the least element of A if for every element a $\in$A, b $\precsim$a.

Note that the least element of a poset is unique if one exists because of the antisymmetry of $\precsim$.

Example 7: The poset of the set of natural numbers with the less-than-or-equal-to relation has the least element 0.

Example 8: The poset of the powerset of {1, 2} with $\subseteq$ has the least element $\emptyset$.

Definition(well order): A total order R on a set A is a well order if every non-empty subset of A has the least element.

Example 9: The poset of the set of natural numbers with the less-than-or-equal-to relation is a well order, because every set of natural numbers has the least element.

The poset of the set of positive real numbers with the less-than-or-equal-to relation is not a well order, because the set itself does not have any least element (0 is not in the set).

A digraph of a binary relation on a set can be simplified if the relation is a partial order. Hasse diagrams defined as follows are such graphs.

Definition(Hasse diagram): A Hasse diagram is a graph for a poset which does not have loops and arcs implied by the transitivity. Further, it is drawn so that all arcs point upward eliminating arrowheads.

To obtain the Hassse diagram of a poset, first remove the loops, then remove arcs $< a, b >$ if and only if there is an element c that $< a, c >$ and $< c, b >$ exist in the given relation.

Example 10: For the relation $\{< a, a >, < a, b >, < a, c >, < b, b >, < b, c >, < c, c >\}$ on set {a, b,c}, the Hasse diagram has the arcs $\{< a, b >, < b, c >\}$ as shown in Figure 10.

7.1.4.3.1 Topological Sorting

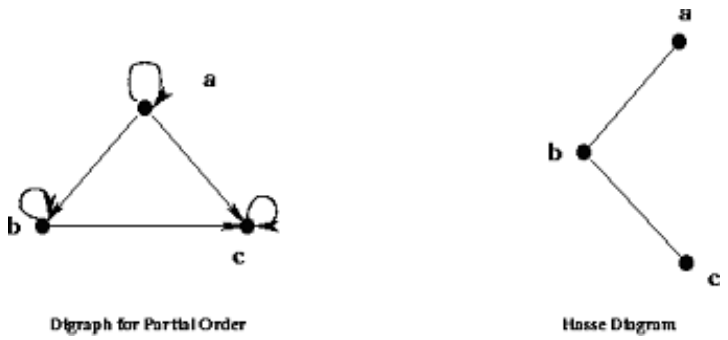


Figure 7.10

The elements in a finite poset can be ordered linearly in a number of ways while preserving the partial order. For example {$\emptyset$, {1}, {2}, {1, 2}} with the partial order $\subseteq$, can be ordered linearly as $\emptyset$, {1}, {2}, {1, 2}, or $\emptyset$, {2}, {1}, {1, 2}. In these orders a set appears before (to the left of) another set if it is a subset of the other. In real life, tasks for manufacturing goods in general can be partially ordered based on the prerequisite relation, that is certain tasks must be completed before certain other tasks can be started. For example the arms of a chair must be carved before the chair is assembled. Scheduling those tasks is essentially the same as arranging them with a linear order (ignoring here some possible concurrent processing for simplicity's sake).

The topological sorting is a procedure to find from a partial order on a finite set a linear order that does not violate the partial order. It is based on the fact that a finite poset has at least one minimal element. The basic idea of the topological sorting is to first remove a minimal element from the given poset, and then repeat that for the resulting set until no more elements are left. The order of removal of the minimal elements gives a linear order. The following algorithm formally describes the topological sorting.

Algorithm Topological Sort

Input: A finite poset <A, R>.

Output: A sequence of the elements of A preserving the order R.

integer i;
i := 1;
while (A $\neq\emptyset$) {
pick a minimal element b from A;
A := A - {b};
i := i + 1;
output b
} Example: Let A = {{1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}} with the partial order $\subseteq$. This given A has three minimal elements {1}, {2}, and {3}.

Select {2} and remove it from A. Let A denote the resultant set i.e. A := A - {2}. The new A has two minimal elements {1}, and {3}.

Select {1} and remove it from A. Denote by A the resultant set, that is A = {{3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}}.

This new A has two minimal elements {3} and {1, 2}.

Select {1, 2} and remove it from A.

Proceeding in like manner, we can obtain the following linear order: {{2}, {1}, {1, 2}, {3}, {1, 3}, {2, 3}, {1, 2, 3}}.

7.1.5 Questions and Exercises

1. Indicate which of the following statements are correct and which are not.
   a. $\ll$1,2>,<1,3$\gg$ is an ordered pair.
   b. Any set of ordered pair is a binary relation.
   c. {<1,2>,<1,3>,<2,3>} is less than relation on {1,2,3}
   d. {1,2} $\times$ {3} = {<1,3>,<2,3>,<3,1>,<3,2>}
2. Indicate which of the following statements are correct and which are not.
   a. Any set of quadruplets is a 4-ary relation.
   b. {<1,2,1>} is a ternary relation on the set of natural numbers.
   c. {1,2} $\times$ {3} $\times$ {4,5} = {1,2} $\times$ {4,5} $\times$ {3}
   d. {<1,2,{1,2}>, <2,3,{2,3}>, <3,1,{1,3}>} is a ternary relation.
3. Indicate which of the following statements are correct and which are not.
   a. {<1,2>,<2,3>} is equal to {<1,2>,<2,3>} as a relation.
   b. The relation {<1,1>,<2,2>} over {1,2} is equal to the relation {<1,1,1>,<2,2,2>} over {1,2}.
   c. The relation {<1,2>,<2,1>} over {1,2} is equal to the relation {<1,2>,<2,1>} over {1,2,3}.
   d. The relation {<1,2>,<2,1>} over {1,2} is equal to the relation {<1,2>,<2,1>,<1,2>} over {1,2}.
4. For the graph in Figure 11, indicate which of the following statements are correct and which are not.

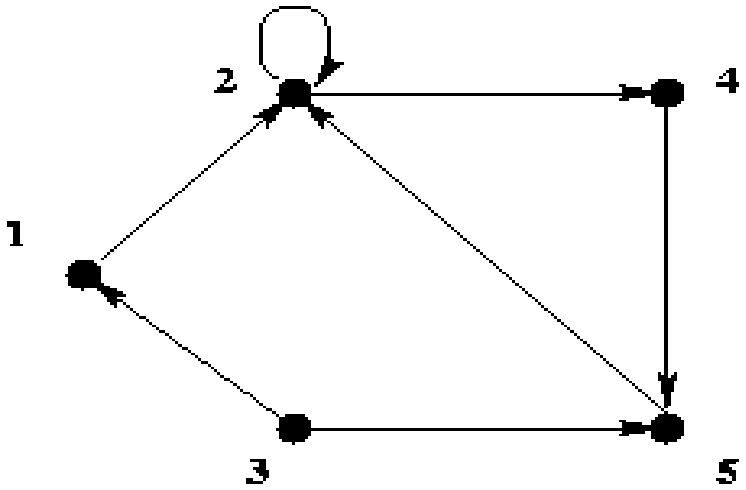


Figure 7.11

a. The in-degree of vertex 2 is 3.
b. Every partial digraph with vertices 2,4 and 5 is connected.
c. 22 is a loop.
d. 213 is a simple directed path.
5. Indicate which of the following statements are correct and which are not.
a. A binary relation on a set can be neither symmetric nor anti-symmetric.
b. For a transitive relation, only vertices connected by a directed path are connected by an arc.
c. If a relation R is symmetric, then for every <a,b> in R <b,a> must be in R.
d. A symmetric relation cannot be transitive an irreflexive at the same time.
6. Indicate which of the following statements are correct and which are not.
a. The intersection of less-than-or-equal-to and greater-than-or-equal-to relations is quality relation.
b. If B is a binary relation on a set A and T is a ternary relation on A, union of B and T is a relation.
c. If <a,b> is in RR, then b can reached from a in 2 or less hops in the digraph of R.
d. The square of less-than-or-equal-to relation is equal to less-than-or-equal-to relation.
7. Indicate which of the following statements are correct and which are not.
a. The symmetric closure of a relation is a relation, and it is symmetric.
b. The transitive closure of a relation contains all the ordered pairs of the relation, and possibly more.
c. If a relation is symmetric, then it is its own symmetric closure.
d. If the digraph of a relation is a simple cycle, then its transitive closure is the universal relation.
8. Indicate which of the following statements are correct and which are not.
a. An equivalence relation must be symmetric.
b. An equivalence relation can be antisymmetric.
c. Objects in different equivalence classes may be related to each other.
d. An equivalence relation is the universal relation on each of its equivalence classes.
9. Indicate which of the following statements are correct and which are not.
a. A total order is a partial order.
b. The partial order can be reconstructed from a Hasse diagram.
c. The reflexive closure of a quasi order is a partial order.
d. Every finite poset has a minimal element and a maximal element

10. List the ordered pairs in the relation **R** from **A** = {0,1, 2, 3} to **B** = {0, 1, 2, 3, 4} where **(a, b)** ∈ **R** if and only if

a. a > b.

b. a + b = 3.

c. a divides **b.**

d. a - b = 0.

e. gcd(**a**, **b**) = **1.**

f. lcm(**a**, **b**) = **6.**

11. Recursively define the relation { (**a**, **b**) | **a=2b** }, where **a** and **b** are natural numbers.

12. List unary relation on {1, 2, 3}.

13. Prove that there are **2**n2 binary relations on a set of cardinality **n**.

14. For each of the following relations on the set {1, 2, 3, 4}, decide whether it is reflexive, symmetric, antisymmetric and/or transitive.

a. {(1, 1), (1, 2), (2, 1), (2, 2), (3, 3), (4, 4)}

b. {(1, 3), (1, 4), (2, 3), (3, 4)}

c. {(1, 1), (1, 3), (1, 4), (2, 1), (2, 3), (2, 4), (3, 1), (3, 3), (3, 4)}

15. Determine whether the relation **R** on the set of all integers is reflexive, symmetric, antisymmetric, and/or transitive, where (**x**, **y**) ∈ **R** if and only if

a. x is divisible by **y**.

b. x ≠ y.

c. y = x + 2 or **y = x - 2.**

d. x = y^2 + 1.

16. Let **A** be the set of people in your town. Let **R**1 be the unary relation representing the people in your town who were registered in the last election and **R**2 be the unary relation representing the people in your town who voted in the last election. Describe the 1-tuples in each of the following relations.

a. R1 ∪**R**2.

b. R1 ∩**R**2.

17. Draw the directed graph that represents the relation {(**a**,**b**), (**a**, **c**), (**b**, **c**), (**c**, **b**), (**c**, **c**), (**c**, **d**), (**d**, **a**), (**d**, **b**)}.

18. Let **R** be the parent-child relation on the set of people that is, **R** = { (**a, b**) | **a** is a parent of **b** }. Let **S** be the sibling relation on the set of people that is, **R** = { (**a**, **b**) | **a** and **b** are siblings (brothers or sisters) }. What are **S** o**R** and **R** o**S**?

19. Let **R** be a reflexive relation on a set **A**. Show that **R** n is reflexive for all positive integers **n**.

20. Let **R** be the relation on the set { 1, 2, 3, 4} containing the ordered pairs (1, 1), (1, 2), (2, 2), (2, 4), (3, 4), and (4, 1). Find

a. the reflexive closure of **R**

b. symmetric closure of **R** and

c. transitive closure of **R**.

21. Let **R** be the relation { (**a, b**) | **a** is a (integer) multiple of **b** } on the set of integers. What is the symmetric closure of **R**?

22. Suppose that a binary relation **R** on a set **A** is reflexive. Show that **R*** is reflexive, where **R*** = $\stackrel{n}{i=1}R^i$.

23. Which of the following relations on {1, 2, 3, 4} are equivalence relations? Determine the properties of an equivalence relation that the others lack.

a. {(1, 1), (2, 2), (3, 3), (4, 4)}

b. {(1, 1), (1, 2), (2, 1), (2, 2), (3, 3), (4, 4)}

c. {(1, 1), (1, 2), (1, 4), (2, 2), (2, 4), (3, 3), (4, 1), (4, 2), (4, 4)}

24. Suppose that **A** is a nonempty set, and **f** is a function that has **A** as its domain. Let **R** be the relation on **A** consisting of all ordered pairs (**x**, **y**) where **f(x) = f(y)**.

a. Show that **R** is an equivalence relation on **A**.

b. What are the equivalence classes of **R**?

25. Show that propositional equivalence is an equivalence relation on the set of all compound propositions.
26. Give a description of each of the congruence classes modulo 6.
27. Which of the following collections of subsets are partitions of {1, 2, 3, 4, 5, 6}?
a. {1, 2, 3}, {3, 4}, {4, 5, 6}
b. {1, 2, 6}, {3, 5}, {4}
c. {2, 4, 6}, {1, 5}
d. {1, 4, 5}, {2, 3, 6}
28. Consider the equivalence relation on the set of integers $\mathbf{R} = \{ (\mathbf{x}, \mathbf{y}) \mid \mathbf{x} - \mathbf{y}$ is an integer$\}$.
a. What is the equivalence class of 1 for this equivalence relation?
b. What is the equivalence class of 0.3 for this equivalence relation?
29. Which of the following are posets?
a. (Z, $=$)
b. (Z, $\neq$)
c. (A collection of sets, $\subseteq$).
30. Draw the Hasse diagram for the divisibility relation on the following sets
a. {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
b. {1, 2, 5, 8, 16, 32}
31. Answer the following questions concerning the poset ({{1}, {2}, {3}, {4}, {1, 3}, {1, 4}, {2, 4}, {3, 4}, {1, 2, 4}, {2, 3, 4}},$\subseteq$).
a. Find the maximal elements.
b. Find the minimal elements.
c. Is there a greatest element?
d Is there a least element?
e. Find all upper bounds of {{2}, {4}}.
f. Find the least upper bound of {{2}, {4}}, if it exists.
g. Find all lower bounds of {{1, 2, 4}, {2, 3, 4}}
h. Find the greatest lower bound of {{1, 2, 4}, {2, 3, 4}}, if it exists.

Chapter 8

Discrete Structures Function[1]

8.1 Function

8.1.1 Definitions on Function

A function is something that associates each element of a set with an element of another set (which may or may not be the same as the first set). The concept of function appears quite often even in non-technical contexts. For example, a social security number uniquely identifies the person, the income tax rate varies depending on the income, and the final letter grade for a course is often determined by test and exam scores, homeworks and projects, and so on.

In all these cases to each member of a set (social security number, income, tuple of test and exam scores, homeworks and projects) some member of another set (person, tax rate, letter grade, respectively) is assigned.

As you might have noticed, a function is quite like a relation. In fact, formally, we define a function as a special type of binary relation.

Definition (function): A function, denote it by f, from a set A to a set B is a relation from A to B that satisfies

1. for each element a in A, there is an element b in B such that <a, b> is in the relation, and
2. if <a, b> and <a, c> are in the relation, then b = c .

The set A in the above definition is called the domain of the function and B its codomain.

Thus, f is a function if it covers the domain (maps every element of the domain) and it is single valued.

The relation given by f between a and b represented by the ordered pair <a, b> is denoted as f(a) = b , and b is called the image of a under f .

The set of images of the elements of a set S under a function f is called the image of the set S under f, and is denoted by f(S) , that is,

f(S) = { f(a) | a ∈ S }, where S is a subset of the domain A of f .

The image of the domain under f is called the range of f.

Example: Let f be the function from the set of natural numbers N to N that maps each natural number x to x2. Then the domain and co-domain of this f are N, the image of, say 3, under this function is 9, and its range is the set of squares, i.e. { 0, 1, 4, 9, 16, ...} .

Definition (sum and product): Let f and g be functions from a set A to the set of real numbers R.

Then the sum and the product of f and g are defined as follows:

For all x, (f + g)(x) = f(x) + g(x) , and

for all x, (f*g)(x) = f(x)*g(x) ,

where f(x)*g(x) is the product of two real numbers f(x) and g(x).

[1] This content is available online at <http://cnx.org/content/m15776/1.1/>.

Example: Let f(x) = 3x + 1 and g(x) = x2 . Then (f + g)(x) = x2 + 3x + 1 , and (f*g)(x) = 3x3 + x2

Definition (one-to-one): A function f is said to be one-to-one (injective) , if and only if whenever f(x) = f(y) , x = y .

Example: The function f(x) = x2 from the set of natural numbers N to N is a one-to-one function. Note that f(x) = x2 is not one-to-one if it is from the set of integers (negative as well as non-negative) to N, because for example f(1) = f(-1) = 1 .

Definition (onto): A function f from a set A to a set B is said to be onto(surjective) , if and only if for every element y of B , there is an element x in A such that f(x) = y , that is, f is onto if and only if f(A) = B .

Example: The function f(x) = 2x from the set of natural numbers N to the set of non-negative even numbers E is an onto function. However, f(x) = 2x from the set of natural numbers N to N is not onto, because, for example, nothing in N can be mapped to 3 by this function.

Definition (bijection): A function is called a bijection , if it is onto and one-to-one.

Example: The function f(x) = 2x from the set of natural numbers N to the set of non-negative even numbers E is one-to-one and onto. Thus it is a bijection.

Every bijection has a function called the inverse function.

These concepts are illustrated in Figure 1. In each figure below, the points on the left are in the domain and the ones on the right are in the co-domain, and arrows show < x, f(x) > relation.

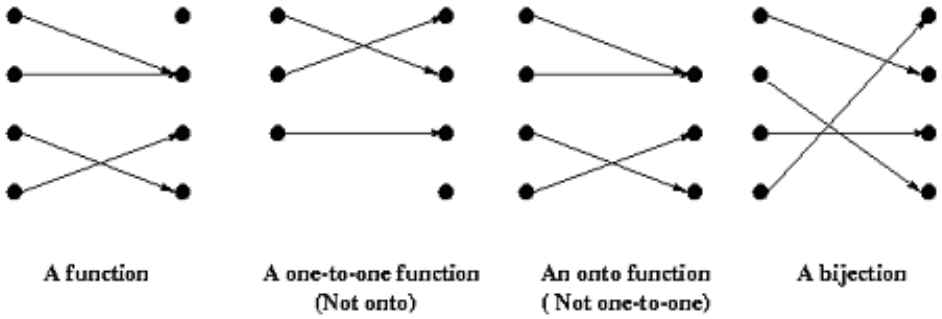


Figure 8.1

Definition (inverse): Let f be a bijection from a set A to a set B. Then the function g is called the inverse function of f, and it is denoted by f -1 , if for every element y of B, g(y) = x , where f(x) = y . Note that such an x is unique for each y because f is a bijection.

For example, the rightmost function in the above figure is a bijection and its inverse is obtained by reversing the direction of each arrow.

Example: The inverse function of f(x) = 2x from the set of natural numbers N to the set of non-negative even numbers E is f -1(x) = 1/2 x from E to N. It is also a bijection.

A function is a relation. Therefore one can also talk about composition of functions.

Definition (composite function): Let g be a function from a set A to a set B, and let f be a function from B to a set C . Then the composition of functions f and g, denoted by fg, is the function from A to C that satisfies

fg(x) = f(g(x)) for all x in A .

Example: Let f(x) = x2 , and g(x) = x + 1 . Then f(g(x)) = (x + 1)2 .

8.1.2 Growth of Functions

8.1.2.1 Introduction

One of the important criteria in evaluating algorithms is the time it takes to complete a job. To have a meaningful comparison of algorithms, the estimate of computation time must be independent of the programming language, compiler, and computer used; must reflect on the size of the problem being solved; and must not depend on specific instances of the problem being solved. The quantities often used for the estimate are the worst case execution time, and average execution time of an algorithm, and they are represented by the number of some key operations executed to perform the required computation.

As an example for an estimate of computation time, let us consider the sequential search algorithm.

Example: Algorithm for Sequential Search

Algorithm SeqSearch(L, n, x)

L is an array with n entries indexed 1, .., n, and x is the key to be searched for in L.

Output: if x is in L , then output its index, else output 0.

index := 1;

while (index $\leq$ n and L[index] $\neq$ x)

index := index + 1 ;

if (index > n) , then index := 0

return index .

The worst case time of this algorithm, for example, can be estimated as follows: First the key operation is comparison of keys comparing L[index] with x . Most search algorithms (if not all) need "comparison of keys". The largest number of execution of this comparison is n , which occurs when x is not in L or when x is at L[n] , and the while loop is executed n times. This quantity n thus obtained is used as an estimate of the worst case time of this sequential search algorithm.

Note that in the while loop two comparisons and one addition are performed. Thus one could use 3n as an estimate just as well. Also note that the very first line and the last two lines are not counted in. The reasons for those are firstly that differences in implementation details such as languages, commands, compilers and machines make differences in constant factors meaningless, and secondly that for large values of n, the highest degree term in n dominates the estimate. Since we are mostly interested in the behavior of algorithms for large values of n , lower terms can be ignored compared with the highest term. The concept that is used to address these issues is something called big-oh, and that is what we are going to study here.

8.1.2.2 Big - Oh

The following example gives the idea of one function growing more rapidly than another. We will use this example to introduce the concept the big-Oh.

Example: f(n) = 100 n2, g(n) = n4, the following table and Figure 2 show that g(n) grows faster than f(n) when n > 10. We say f is big-Oh of g.

n	f(n)	g(n)
10	10,000	10,000
50	250,000	6,250,000
100	1,000,000	100,000,000
150	2,250,000	506,250,000

Table 8.1

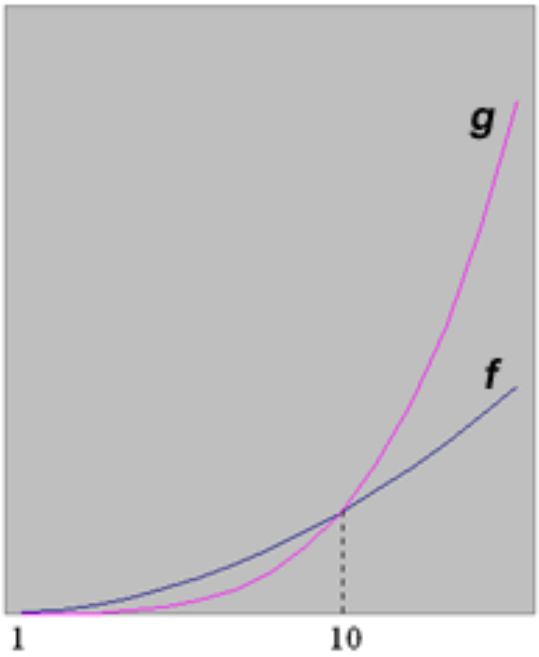


Figure 8.2

Definition (big-oh): Let f and g be functions from the set of integers (or the set of real numbers) to the set of real numbers. Then f(x) is said to be O(g(x)) , which is read as f(x) is big-oh of g(x) , if and only if there are constants C and n0 such that

$| f(x) | \leq C | g(x) |$

whenever $x > n0$.

Note that big-oh is a binary relation on a set of functions (What kinds of properties does it have ? reflexive ? symmetric ? transitive ?).

The relationship between f and g can be illustrated as follows when f is big-oh of g.

For example, $5 x + 10$ is big-oh of $x2$, because $5 x + 10 < 5 x2 + 10 x2 = 15 x2$ for $x > 1$.

Hence for $C = 15$ and $n0 = 1$, $| 5x + 10 | \leq C | x2 |$. Similarly it can be seen that $3 x2 + 2 x + 4 < 9 x2$ for $x > 1$. Hence $3 x2 + 2 x + 4$ is O(x2) . In general, we have the following theorem:

Theorem 1: an xn + ... + a1 x + a0 is O(xn) for any real numbers an , ..., a0 and any nonnegative number n .

Note: Let $f(x) = 3 x2 + 2 x + 4$, $g(x) = x2$, from the above illustration, we have that f(x) is O(g(x)). Also, since $x2 < 3 x2 + 2 x + 4$, we can also get g(x) is O(f(x)). In this case, we say these two functions are of the same order.

8.1.2.3 Growth of Combinations of Functions

Big-oh has some useful properties. Some of them are listed as theorems here. Let use start with the definition of max function.

Definition (max function): Let f1(x) and f2(x) be functions from a set A to a set of real numbers B. Then max(f1(x) , f2(x)) is the function from A to B that takes as its value at each point x the larger of f1(x) and f2(x).

Theorem 2: If f1(x) is O(g1(x)) , and f2(x) is O(g2(x)) , then (f1 + f2)(x) is O(max(g1(x) , g2(x))) .

From this theorem it follows that if f1(x) and f2(x) are O(g(x)) , then (f1 + f2)(x) is O(g(x)) , and

(f1 + f2)(x) is O(max(f1(x) , f2(x))) .

Theorem 3: If f1(x) is O(g1(x)) , and f2(x) is O(g2(x)) , then (f1 * f2)(x) is O(g1(x) * g2(x)) .

8.1.2.4 Big - Omega and Big - Theta

Big-oh concerns with the "less than or equal to" relation between functions for large values of the variable. It is also possible to consider the "greater than or equal to" relation and "equal to" relation in a similar way. Big-Omega is for the former and big-theta is for the latter.

Definition (big-omega): Let f and g be functions from the set of integers (or the set of real numbers) to the set of real numbers. Then f(x) is said to be $\Omega(g(x))$, which is read as f(x) is big-omega of g(x) , if there are constants C and n0 such that

$| f(x) | \geq C | g(x) |$

whenever $x > n0$.

Definition (big-theta): Let f and g be functions from the set of integers (or the set of real numbers) to the set of real numbers. Then f(x) is said to be θ(g(x)) , which is read as f(x) is big-theta of g(x) , if f(x) is O(g(x)), and Ω(g(x)) . We also say that f(x) is of order g(x) .

For example, 3x2 - 3x - 5 is Ω(x2) , because 3x2 - 3x - 5 $\geq$x2 for integers $x > 2$ ($C = 1$, $n0 = 2$) .

Hence by Theorem 1 it is θ(x2) .

In general, we have the following theorem:

Theorem 4: an xn + ... + a1 x + a0 is θ(xn) for any real numbers an , ..., a0 and any nonnegative number n .

8.1.2.5 Little - Oh and Little - Omega

If f(x) is O(g(x)), but not θ(g(x)) , then f(x) is said to be o(g(x)) , and it is read as f(x) is little-oh of g(x) . Similarly for little-omega (ω).

For example x is o(x2) , x2 is o(2x) , 2x is o(x !) , etc.

8.1.3 Calculation of Big – Oh

Basic knowledge of limits and derivatives of functions from calculus is necessary here. Big-oh relationships between functions can be tested using limit of function as follows:

Let f(x) and g(x) be functions from a set of real numbers to a set of real numbers.

Then

1. If $\lim_{x\to\infty} f(x)/g(x) = 0$, then f(x) is o(g(x)) . Note that if f(x) is o(g(x)), then f(x) is O(g(x)).
2. If $\lim_{x\to\infty} f(x)/g(x) = \infty$, then g(x) is o(f(x)) .
3. If $0 < \lim_{x\to\infty} f(x)/g(x) < \infty$, then f(x) is θ(g(x)) .
4. If $\lim_{x\to\infty} f(x)/g(x) < \infty$, then f(x) is O(g(x)) .

For example,

$\lim_{x\to\infty}$ (4x3 + 3x2 + 5)/(x4 − 3x3 − 5x -4)

= $\lim_{x\to\infty}$ (4/x + 3/x2 + 5/x4)/(1 - 3/x - 5/x3 - 4/x4) = 0 .

Hence

(4x3 + 3x2 + 5) is o(x4 - 3x3 - 5x - 4),

or equivalently, (x4 - 3x3 - 5x - 4) is ω(4x3 + 3x2 + 5) .

Let us see why these rules hold. Here we give a proof for 4. Others can be proven similarly.

Proof: Suppose $\lim_{x\to\infty} f(x)/g(x) = C_1 < \infty$.

By the definition of limit this means that

$\forall\varepsilon > 0$, $\exists$n0 such that $|f(x)/g(x) - C1| < \varepsilon$ whenever $x > n0$
Hence $-\varepsilon < f(x)/g(x) - C1 < \varepsilon$
Hence $-\varepsilon + C1 < f(x)/g(x) < \varepsilon + C1$
In particular $f(x)/g(x) < \varepsilon + C1$
Hence $f(x) < (\varepsilon + C1)g(x)$
Let $C = \varepsilon + C1$, then $f(x) < Cg(x)$ whenever $x > n0$.
Since we are interested in non-negative functions f and g, this means that $|f(x)| \leq C | g(x)|$
Hence $f(x) = O(g(x))$.

8.1.3.1 L'Hospital (L'Hôpital)'s Rule

$\lim_{x\to\infty} f(x)/g(x)$ is not always easy to calculate. For example take $\lim_{x\to\infty} x2/3x$. Since both x2 and 3x go to ∞ as x goes to ∞ and there is no apparent factor common to both, the calculation of the limit is not immediate. One tool we may be able to use in such cases is L'Hospital's Rule, which is given as a theorem below.

8.1.3.1.1 Theorem 5 (L'Hospital):

If $\lim_{x\to\infty} f(x) = \infty$ and $\lim_{x\to\infty} g(x) = \infty$, and f(x) and g(x) have the first derivatives, f '(x) and g'(x) , respectively, then $\lim_{x\to\infty} f(x)/g(x) = f'(x)/g'(x)$.

This also holds when $\lim_{x\to\infty} f(x) = 0$ and $\lim_{x\to\infty} g(x) = 0$, instead of $\lim_{x\to\infty} f(x) = \infty$ and $\lim_{x\to\infty} g(x) = \infty$.

For example, $\lim_{x\to\infty} x/ex = \lim_{x\to\infty} 1/ex = 0$, because (ex)' = ex, where e is the base for the natural logarithm.

Similarly $\lim_{x\to\infty} \ln x/x = (1/x)/1 = \lim_{x\to\infty} 1/x = 0$.

Note that this rule can be applied repeatedly as long as the conditions are satisfied.

So, for example, $\lim_{x\to\infty} x2/ex = \lim_{x\to\infty} 2x/ex = \lim_{x\to\infty} 2/ex = 0$.

8.1.4 Summary of Big – Oh

Sometimes, it is very necessary to compare the order of some common used functions including the following:

1	logn	n	nlogn	n2	2n	n!	nn

Table 8.2

Now, we can use what we've learned above about the concept of big-Oh and the calculation methods to calculate the order of these functions. The result shows that each function in the above list is big-oh of the functions following them. Figure 2 displays the graphs of these functions, using a scale for the values of the functions that doubles for each successive marking on the graph.

SORRY, THIS MEDIA TYPE IS NOT SUPPORTED.

8.1.5 Questions and Exercises

1. Indicate which of the following statements are correct and which are not.
   a. The range of a function is a subset of the co-domain.
   b. The cardinality of the domain of a function is not less than that of its range.
   c. The range of a function is the image of its domain.
   d. Max {f,g} is the function that takes as its value at x the larger of f(x) and g(x).
2. Indicate which of the following statements are correct and which are not.
   a. $\lim_{x\to\infty}$ (n2 + 3n + 5)/(4n2 + 10n + 6) = 1/4.
   b. 2n is big-theta of 3n.
   c. $\lim_{x\to\infty}$ (10n3 + 3n2 + 500n + 100)/(2n4 + 3n3) = $\lim_{x\to\infty}$ (6n + 6)/(24n2 + 18n)
   d. $\lim_{x\to\infty}$ f'/g' = $\lim_{x\to\infty}$ f"/g". if f" and g" exist, and $\lim_{x\to\infty}$ f' and $\lim_{x\to\infty}$ g' are both equal to infinity or 0.
3. Which **f** is not a function from R to R in the following equations, where R is the set of real numbers? Explain why they are not a function.
   a. f(x) = 1/x
   b. f(x) = y such that **y** 2 = **x**
   c. f(x) = x 2 − **1**
4. Find the domain and range of the following functions.
   a. the function that assigns to each bit string (of various lengths) the number of zeros in it.
   b. the function that assigns the number of bits left over when a bit string (of various lengths) is split into bytes (which are blocks of 8 bits)
5. Determine whether each of the following functions from Z to Z is one-to-one, where Z is the set of integers.
   a. f(n) = n + 2
   b. f(n) = n^2 + n + 1
   c. f(n) = n^3 - 1
6. Determine whether each of the following functions from Z to Z is onto.
   a. f(n) = n + 2
   b. f(n) = n^2 + n + 1
   c. f(n) = n^3 - 1
7. Determine whether each of the following functions is a bijection from R to R.
   a. f(x) = 2x + 3
   b. f(x) = x^2 + 2
8. Determine whether each of the following functions from R to R is **O(x)**.
   a. f(x) = 10
   b. f(x) = 3x + 7
   c. f(x) = x^2 + x + 1

d. f(x) = 5ln x

9. Use the definition of big-oh to show that $x4 + 5x3 + 3x2 + 4x + 6$ is $O(x4)$.
10. Show that $(x^2 + 2x + 3) / (x + 1)$ is $O(x)$.
11. Show that $5x4 + x^2 + 1$ is $O(x4/2)$ and $x4/2$ is $O(5x4 + x^2 + 1)$.
12. Show that 2n is $O(3n)$ but that 3n is not $O(2n)$.
13. Explain what it means for a function to be $O(1)$.
14. Give as good (i.e. small) a big-O estimate as possible for each of the following functions.

a. $(n^2 + 3n + 8)(n + 1)$
b. $(3\log n + 5n^2)(n^3 + 3n + 2)$

Attributions

Collection: *Discrete Structures*
Edited by: Duy Bui
URL: http://cnx.org/content/col10513/1.1/
License: http://creativecommons.org/licenses/by/2.0/

Module: "Discrete Structures Course Information"
By: Duy Bui
URL: http://cnx.org/content/m14585/1.3/
Pages: 1-14
Copyright: Duy Bui
License: http://creativecommons.org/licenses/by/2.0/

Module: "Discrete Structures Introduction"
By: Duy Bui
URL: http://cnx.org/content/m14586/1.3/
Pages: 15-17
Copyright: Duy Bui
License: http://creativecommons.org/licenses/by/2.0/

Module: "Discrete Structures Problem Solving"
By: Duy Bui
URL: http://cnx.org/content/m15771/1.1/
Pages: 19-31
Copyright: Duy Bui
License: http://creativecommons.org/licenses/by/2.0/

Module: "Discrete Structures Logic"
By: Duy Bui
URL: http://cnx.org/content/m15773/1.1/
Pages: 33-62
Copyright: Duy Bui
License: http://creativecommons.org/licenses/by/2.0/

Module: "Discrete Structures Set Theory"
By: Duy Bui
URL: http://cnx.org/content/m15772/1.1/
Pages: 63-70
Copyright: Duy Bui
License: http://creativecommons.org/licenses/by/2.0/

Module: "Discrete Structures Recursion"
By: Duy Bui
URL: http://cnx.org/content/m15774/1.1/
Pages: 71-80
Copyright: Duy Bui
License: http://creativecommons.org/licenses/by/2.0/

Module: "Discrete Structures Relation"
By: Duy Bui
URL: http://cnx.org/content/m15775/1.1/
Pages: 81-99

Module: "Discrete Structures Function"
By: Duy Bui
URL: http://cnx.org/content/m15776/1.1/
Pages: 101-108

Discrete Structures
This course covers the fundamental mathematical concepts and reasoning along with problem solving techniques. Topics covered include propositional logic, predicate logic, inferencing, proof methods including induction, set operations, binary relations including order relations, and equivalence relations, graphs, and functions.

About Connexions
Since 1999, Connexions has been pioneering a global system where anyone can create course materials and make them fully accessible and easily reusable free of charge. We are a Web-based authoring, teaching and learning environment open to anyone interested in education, including students, teachers, professors and lifelong learners. We connect ideas and facilitate educational communities.

Connexions's modular, interactive courses are in use worldwide by universities, community colleges, K-12 schools, distance learners, and lifelong learners. Connexions materials are in many languages, including English, Spanish, Chinese, Japanese, Italian, Vietnamese, French, Portuguese, and Thai. Connexions is part of an exciting new information distribution system that allows for **Print on Demand Books**. Connexions has partnered with innovative on-demand publisher QOOP to accelerate the delivery of printed course materials and textbooks into classrooms worldwide at lower prices than traditional academic publishers.