BBM105

CREDIT TO: *SHANNON I. STEINFADT*

INVITATION TO COMPUTER SCIENCE

ALGORITHM DISCOVERY AND DESIGN

Objectives

In this chapter, you will learn about

- Representing algorithms
- Examples of algorithmic problem solving

Introduction

- *A procedure for solving a mathematical problem in a finite number of steps that frequently involves repetition of an operation;*
- *Broadly: a step-by-step method for accomplishing some task.*

Abu Ja' far Muhammad ibn Musa Al-Khowarizmi (a.d. 780–850?)

The word *algorithm* is derived from the last name of Muhammad ibn Musa Al-Khowarizmi, a famous Persian mathematician and author from the eighth and ninth centuries. Al-Khowarizmi was a teacher at the Mathematical Institute in Baghdad and the author of the book *Kitab al jabr w'al muqabala*, which in English means "Rules of Restoration and Reduction." It is one of the earliest mathematical textbooks, and its title gives us the word *algebra* (the Arabic word *al jabr* means "reduction").

In 825 A.D., Al-Khowarizmi wrote another book about the base-10 positional numbering system that had recently been developed in India. In this book he described formalized, step-by-step procedures for doing arithmetic operations, such as addition, subtraction, and multiplication, on numbers represented in this new decimal system. In the twelfth century this book was translated into Latin, introducing the base-10 Hindu-Arabic numbering system to Europe, and Al-Khowarizmi's name became closely associated with these formal numerical techniques. His last name was rendered as Algorismus in Latin characters, and eventually the formalized procedures that he pioneered and developed became known as *algorithms* in his honor.

Representing Algorithms

- Natural language
  - Language spoken and written in everyday life
  - Examples: English, Spanish, Arabic, and so on
  - Problems with using natural language for algorithms
    - Verbose
    - Imprecise
    - Relies on context and experiences to give precise meaning to a word or phrase

Adding Two m-Digit Numbers

Initially, set the value of the variable *carry* to 0 and the value of the variable i to 0. When these initializations have been completed, begin looping as long as the value of the variable i is less than or equal to $(m - 1)$. First, add together the values of the two digits a_i and b_i and the current value of the carry digit to get the result called c_i. Now check the value of c_i to see whether it is greater than or equal to 10. If c_i is greater than or equal to 10, then reset the value of *carry* to 1 and reduce the value of c_i by 10; otherwise, set the value of *carry* to zero. When you are done with that operation, add 1 to i and begin the loop all over again. When the loop has completed execution, set the leftmost digit of the result c_m to the value of *carry* and print out the final result, which consists of the digits $c_m\ c_m - 1 \ldots c_0$. After printing the result, the algorithm is finished, and it terminates.

Figure 2.1

The Addition Algorithm of Figure 1.2 Expressed in Natural Language

Representing Algorithms (continued)

- High-level programming language
  - Examples: C++, Java, Python
  - Problem with using a high-level programming language for algorithms
    - During the initial phases of design, we are forced to deal with detailed language issues

```
{
int i, m, Carry;
int[] a = new int[100];
int[] b = new int[100];
int[] c = new int[100];
m = Console.readInt();
for (int j = 0; j < =m−1; j++)    {
          a[j] = Console.readInt();
          b[j] = Console.readInt();
}
Carry = 0;
i = 0;
while (i < m)    {
          c[i] = a[i] + b[i] + Carry;
          if (c[i] > =10)
          .
          .
          .
```

Figure 2.2

The Beginning of the Addition Algorithm of Figure 1.2 Expressed in a High-Level Programming Language

Algorithm for Adding Two m-Digit Numbers

Given: $m \geq 1$ and two positive numbers each containing m digits, $a_{m-1}\, a_{m-2} \ldots a_0$ and $b_{m-1}\, b_{m-2} \ldots b_0$
Wanted: $c_m c_{m-1}\, c_{m-2} \ldots c_0$, where $c_m c_{m-1}\, c_{m-2} \ldots c_0 = (a_{m-1}\, a_{m-2} \ldots a_0) + (b_{m-1}\, b_{m-2} \ldots b_0)$

Algorithm:

Step 1 Set the value of *carry* to 0.
Step 2 Set the value of i to 0.
Step 3 While the value of i is less than or equal to $m - 1$, repeat the instructions in steps 4 through 6.
Step 4 Add the two digits a_i and b_i to the current value of *carry* to get c_i.
Step 5 If $c_i \geq 10$, then reset c_i to $(c_i - 10)$ and reset the value of *carry* to 1; otherwise, set the new value of *carry* to 0.
Step 6 Add 1 to i, effectively moving one column to the left.
Step 7 Set c_m to the value of *carry*.
Step 8 Print out the final answer, $c_m\, c_{m-1}\, c_{m-2} \ldots c_0$.
Step 9 Stop.

Figure 1.2

Pseudocode

- English language constructs modeled to look like statements available in most programming languages
- Steps presented in a structured manner (numbered, indented, and so on)
- No fixed syntax for most operations is required

Pseudocode (continued)

- Less ambiguous and more readable than natural language
- Emphasis is on process, not notation
- Well-understood forms allow logical reasoning about algorithm behavior
- Can be easily translated into a programming language

Sequential Operations

- Types of algorithmic operations
  - Sequential
  - Conditional
  - Iterative

Sequential Operations (continued)

- Computation operations
  - Example
    - Set the value of "variable" to "arithmetic expression"
  - Variable
    - Named storage location that can hold a data value

Sequential Operations (continued)

- Input operations
  - To receive data values from the outside world
  - Example
    - Get a value for *r*, the radius of the circle
- Output operations
  - To send results to the outside world for display
  - Example
    - Print the value of *Area*

Average Miles per Gallon Algorithm (Version 1)

Step	Operation
1	Get values for *gallons used, starting mileage, ending mileage*
2	Set value of *distance driven* to (*ending mileage – starting mileage*)
3	Set value of *average miles per gallon* to (*distance driven ÷ gallons used*)
4	Print the value of *average miles per gallon*
5	Stop

Figure 2.3

Algorithm for Computing Average Miles per Gallon

Conditional and Iterative Operations

- Sequential algorithm
  - Also called straight-line algorithm
  - Executes its instructions in a straight line from top to bottom and then stops
- Control operations
  - Conditional operations
  - Iterative operations

Conditional and Iterative Operations (continued)

- Conditional operations
  - Ask questions and choose alternative actions based on the answers
  - Example
    - if x is greater than 25 then

      print x

      else

      print x times 100

Conditional and Iterative Operations (continued)

- Iterative operations
  - Perform “looping” behavior, repeating actions until a continuation condition becomes false
  - Loop
    - The repetition of a block of instructions

Conditional and Iterative Operations (continued)

- Examples
  - while $j > 0$ do

    set s to $s + a_j$

    set j to $j - 1$

  - repeat

    print a_k

    set k to $k + 1$

    until $k > n$

Average Miles per Gallon Algorithm (Version 2)

Step	Operation
1	Get values for *gallons used*, *starting mileage*, *ending mileage*
2	Set value of *distance driven* to (*ending mileage* – *starting mileage*)
3	Set value of *average miles per gallon* to (*distance driven* ÷ *gallons used*)
4	Print the value of *average miles per gallon*
5	If *average miles per gallon* is greater than 25.0 then
6	Print the message 'You are getting good gas mileage'
	Else
7	Print the message 'You are NOT getting good gas mileage'
8	Stop

Figure 2.5
Second Version of the Average Miles per Gallon Algorithm

Conditional and Iterative Operations (continued)

- Components of a loop
  - Continuation condition
  - Loop body
- Infinite loop
  - The continuation condition never becomes false
  - An error

Average Miles per Gallon Algorithm (Version 3)

Step	Operation
1	*response* = Yes
2	While (*response* = Yes) do steps 3 through 11
3	Get values for *gallons used*, *starting mileage*, *ending mileage*
4	Set value of *distance driven* to (*ending mileage* – *starting mileage*)
5	Set value of *average miles per gallon* to (*distance driven* ÷ *gallons used*)
6	Print the value of *average miles per gallon*
7	If average miles per gallon > 25.0 then
8	Print the message 'You are getting good gas mileage'
	Else
9	Print the message 'You are NOT getting good gas mileage'
10	Print the message 'Do you want to do this again? Enter Yes or No'
11	Get a new value for *response* from the user
12	Stop

Figure 2.7

Third Version of the Average Miles per Gallon Algorithm

Conditional and Iterative Operations (continued)

- Pretest loop
  - Continuation condition tested at the beginning of each pass through the loop
  - It is possible for the loop body to never be executed
  - While loop

Conditional and Iterative Operations (continued)

- Posttest loop
  - Continuation condition tested at the end of loop body
  - Loop body must be executed at least once
  - Do/While loop

FIGURE 2.6

Summary of Pseudocode Language Instructions

COMPUTATION:

```
Set the value of "variable" to "arithmetic expression"
```

INPUT/OUTPUT:

```
Get a value for "variable", "variable". . .
Print the value of "variable", "variable", . . .
Print the message 'message'
```

CONDITIONAL:

```
If "a true/false condition" is true then
    first set of algorithmic operations
Else
    second set of algorithmic operations
```

ITERATIVE:

```
While ("a true/false condition") do step i through step j
        Step i: operation
        .
        .
        .
        Step j: operation

While ("a true/false condition") do
        operation
        .
        .
        .
        operation
End of the loop

Do
        operation
        operation
        .
        .
        .
While ("a true/false condition")
```

Figure 2.9

Summary of Pseudocode Language Instructions

Examples of Algorithmic Problem Solving

- Go Forth and Multiply: Multiply two numbers using repeated addition
- Sequential search: Find a particular value in an unordered collection
- Find maximum: Find the largest value in a collection of data
- Pattern matching: Determine if and where a particular pattern occurs in a piece of text

Example 1: Go Forth and Multiply

- Task
  - Implement an algorithm to multiply two numbers, *a* and *b*, using repeated addition
- Algorithm outline
  - Create a loop that executes exactly *b* times, with each execution of the loop adding the value of *a* to a running total

Multiplication via Repeated Addition

```
Get values for a and b
If (either a = 0 or b = 0) then
    Set the value of product to 0
Else
    Set the value of count to 0
    Set the value of product to 0
    While (count < b) do
        Set the value of product to (product + a)
        Set the value of count to (count+1)
    End of loop
Print the value of product
Stop
```

Figure 2.10

Algorithm for Multiplication via Repeated Addition

Example 2: Looking, Looking, Looking

- Task
  - Find a particular person's name from an unordered list of telephone subscribers
- Algorithm outline
  - Start with the first entry and check its name, then repeat the process for all entries

Example 2: Looking, Looking, Looking (continued)

- Algorithm discovery
  - Finding a solution to a given problem
- Naïve sequential search algorithm
  - For each entry, write a separate section of the algorithm that checks for a match
  - Problems
    - Only works for collections of exactly one size
    - Duplicates the same operations over and over

Example 2: Looking, Looking, Looking (continued)

- Correct sequential search algorithm
  - Uses iteration to simplify the task
  - Refers to a value in the list using an index (or pointer)
  - Handles special cases (such as a name not found in the collection)
  - Uses the variable *Found* to exit the iteration as soon as a match is found

Sequential Search Algorithm

Step	Operation
1	Get values for *NAME*, $N_1, \ldots, N_{10,000}$, and $T_1, \ldots, T_{10,000}$
2	Set the value of *i* to 1 and set the value of *Found* to NO
3	While both (*Found* = NO) and ($i \leq 10{,}000$) do steps 4 through 7
4	If *NAME* is equal to the *i*th name on the list N_i then
5	Print the telephone number of that person, T_i
6	Set the value of *Found* to YES
	Else (*NAME* is not equal to N_i)
7	Add 1 to the value of *i*
8	If (*Found* = NO) then
9	Print the message 'Sorry, this name is not in the directory'
10	Stop

Figure 2.13

The Sequential Search Algorithm

Example 2: Looking, Looking, Looking (continued)

- The selection of an algorithm to solve a problem is greatly influenced by the way the data for that problem is organized

Example 3: Big, Bigger, Biggest

- Task
  - Find the largest value from a list of values
- Algorithm outline
  - Keep track of the largest value seen so far (initialized to be the first in the list)
  - Compare each value to the largest seen so far, and keep the larger as the new largest

Example 3: Big, Bigger, Biggest (continued)

- Once an algorithm has been developed, it may itself be used in the construction of other, more complex algorithms
- Library
  - A collection of useful algorithms
  - An important tool in algorithm design and development

Example 3: Big, Bigger, Biggest (continued)

- Find Largest algorithm
  - Uses iteration and indices as in previous example
  - Updates *location* and *largest so far* when needed in the loop

Find Largest Algorithm

Get a value for n, the size of the list
Get values for $A_1, A_2, \ldots, A_n$, the list to be searched
Set the value of *largest so far* to A_1
Set the value of *location* to 1
Set the value of i to 2
While ($i \leq n$) do
    If $A_i >$ *largest so far* then
        Set *largest so far* to A_i
        Set *location* to i
    Add 1 to the value of i
End of the loop
Print out the values of *largest so far* and *location*
Stop

Figure 2.14
Algorithm to Find the Largest Value in a List

Example 4: Meeting Your Match

- Task
  - Find if and where a pattern string occurs within a longer piece of text
- Algorithm outline
  - Try each possible location of pattern string in turn
  - At each location, compare pattern characters against string characters

Example 4: Meeting Your Match (continued)

- Abstraction
  - Separating high-level view from low-level details
  - Key concept in computer science
  - Makes difficult problems intellectually manageable
  - Allows piece-by-piece development of algorithms

Example 4: Meeting Your Match (continued)

- Top-down design
  - When solving a complex problem
    - Create high-level operations in the first draft of an algorithm
    - After drafting the outline of the algorithm, return to the high-level operations and elaborate each one
    - Repeat until all operations are primitives

Example 4: Meeting Your Match (continued)

- Pattern-matching algorithm
  - Contains a loop within a loop
    - External loop iterates through possible locations of matches to pattern
    - Internal loop iterates through corresponding characters of pattern and string to evaluate match

Pattern-Matching Algorithm

Get values for n and m, the size of the text and the pattern, respectively
Get values for both the text $T_1\ T_2 \ldots T_n$ and the pattern $P_1\ P_2 \ldots P_m$
Set k, the starting location for the attempted match, to 1
While $(k \leq (n - m + 1))$ do
    Set the value of i to 1
    Set the value of *Mismatch* to NO
    While both $(i \leq m)$ and (*Mismatch* = NO) do
        If $P_i \neq T_{k+(i-1)}$ then
            Set *Mismatch* to YES
        Else
            Increment i by 1 (to move to the next character)
    End of the loop
    If *Mismatch* = NO then
        Print the message 'There is a match at position'
        Print the value of k
    Increment k by 1
End of the loop
Stop, we are finished

Figure 2.16

Final Draft of the Pattern-Matching Algorithm

Summary

- Algorithm design is a first step in developing an algorithm
- Algorithm design must
  - Ensure the algorithm is correct
  - Ensure the algorithm is sufficiently efficient
- Pseudocode is used to design and represent algorithms

Summary

- Pseudocode is readable, unambiguous, and able to be analyzed
- Algorithm design is a creative process; uses multiple drafts and top-down design to develop the best solution
- Abstraction is a key tool for good design

ENTITY-RELATIONSHIP MODEL

All slides credit to: Shakila Mahjabin

E- R DATA MODELING

- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- Entities have **attributes**
  - Example: people have *names* and *addresses*
  -
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays

ATTRIBUTES

- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.
  - Example:

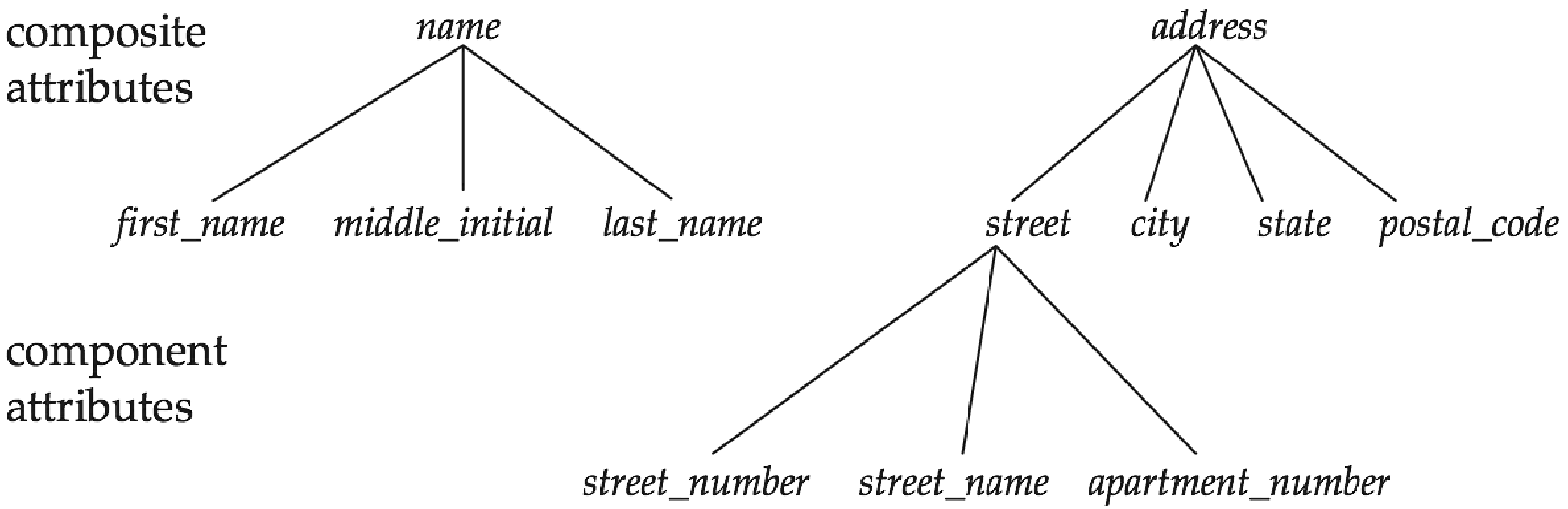
    instructor = (*ID, name, street, city, salary*)
    course= (*course_id, title, credits*)
- **Domain** – the set of permitted values for each attribute

- Attribute types:
  - **Simple** and **composite** attributes.
  - **Single-valued** and **multivalued** attributes
  - **Derived** attributes

TYPES OF ATTRIBUTES

Simple Attribute: Attribute that consist of a single atomic value.
Example: Salary

Composite Attribute : Attribute value not atomic.
Example : Address : ‘House _no:City:State
Name : ‘First Name: Middle Name: Last Name’



TYPES OF ATTRIBUTES

Single Valued Attribute: Attribute that hold a single value

Exampe1: City
Example2:Customer id

Multi Valued Attribute: Attribute that hold multiple values.

Example1: A customer can have multiple phone numbers, email id's etc
Example2: A person may have several college degrees

Derived Attribute: An attribute that's value is derived from a stored attribute.

Example : age, and it's value is derived from the stored attribute Date of Birth.

Entity Sets *instructor* and *student*

instructor_ID	instructor_name
76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

instructor

student-ID	student_name
98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

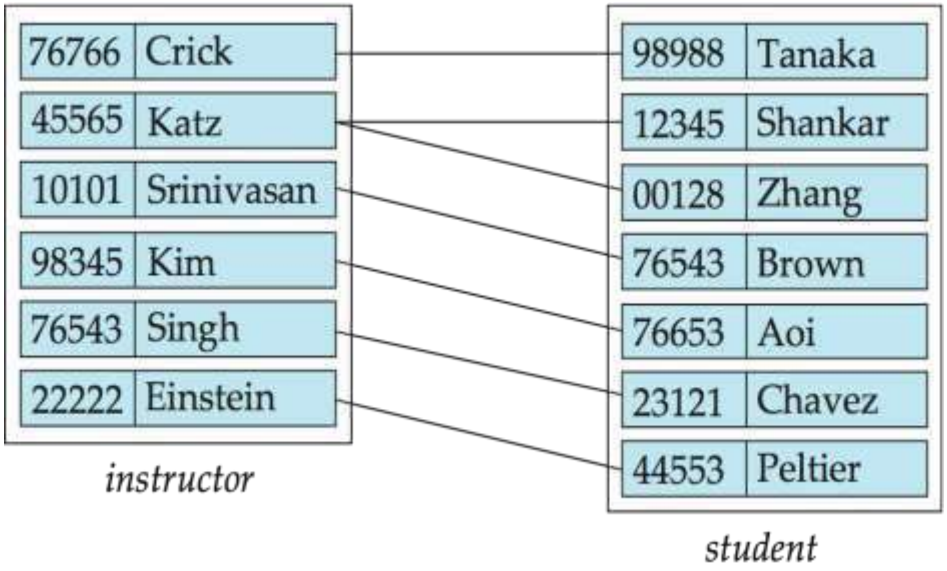
student

RELATIONSHIP SETS

- A **relationship** is an association among several entities

  Example:

  44553 (Peltier) — *advisor* — 22222 (Einstein)
  student entity — relationship set — *instructor* entity

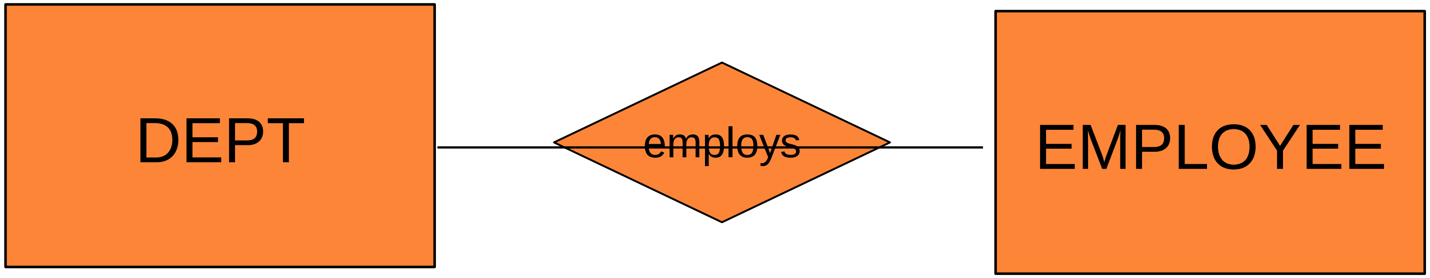


Entity-Relationship Diagrams

- Representing entities
  - we represent an entity by a named rectangle
  - use a singular noun, or adjective + noun
  - refer to one instance in naming

Entity-Relationship Diagrams

- Representing relationship

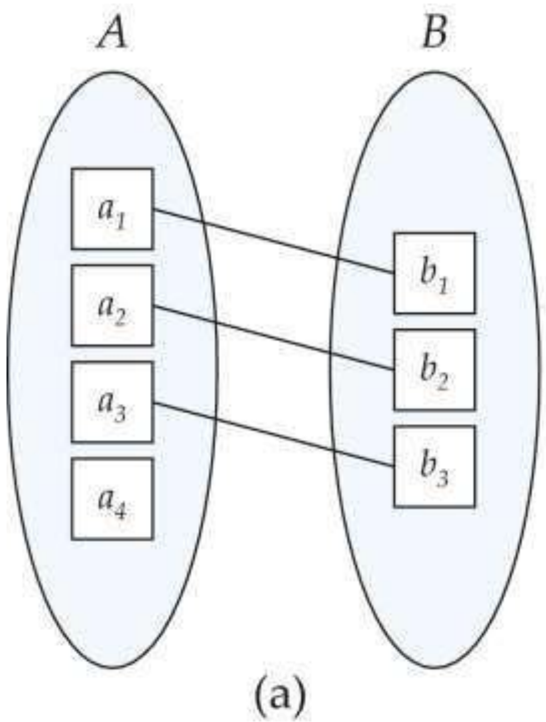


Entity-Relationship Diagrams

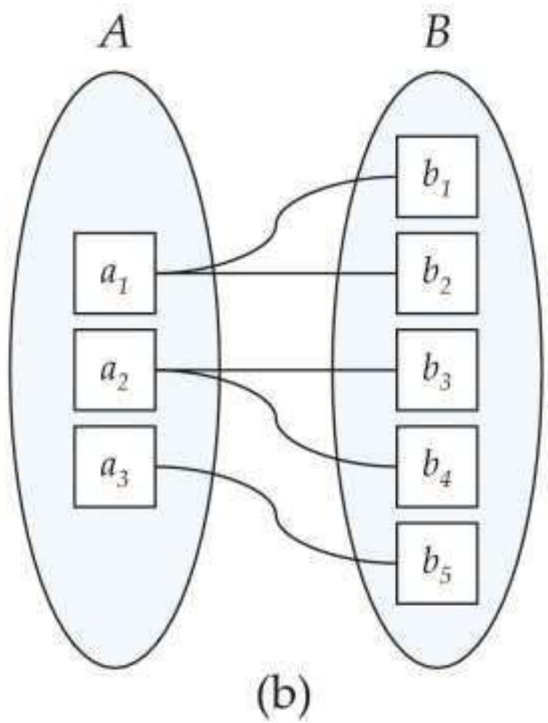
Types of Relationships

- Three types of relationships can exist between entities
- One-to-one relationship (1:1): One instance in an entity (parent) refers to one and only one instance in the related entity (child).
- One-to-many relationship (1:M): One instance in an entity (parent) refers to one or more instances in the related entity (child)

Entity-Relationship Diagrams



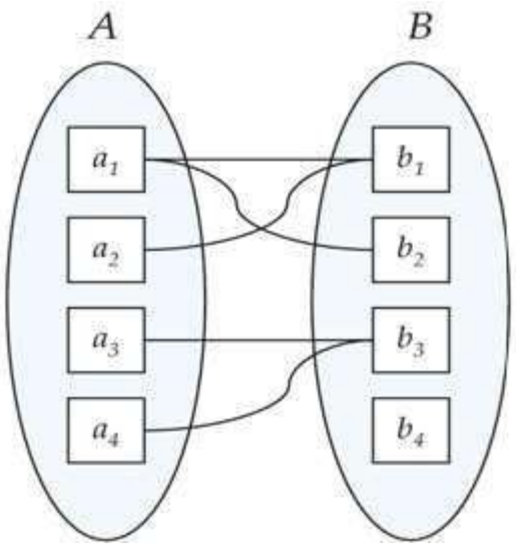
One to one



One to many

Entity-Relationship Diagrams

- Types of Relationships
  - Many-to-many relationship (M:N): exists when one instance of the first entity (parent) can relate to many instances of the second entity (child), and one instance of the second entity can relate to many instances of the first entity.



Many to many

Cardinality Constraints

- We express cardinality constraints by drawing either a directed line (→), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.
- Or, by numbering each entity. * or, m for many.

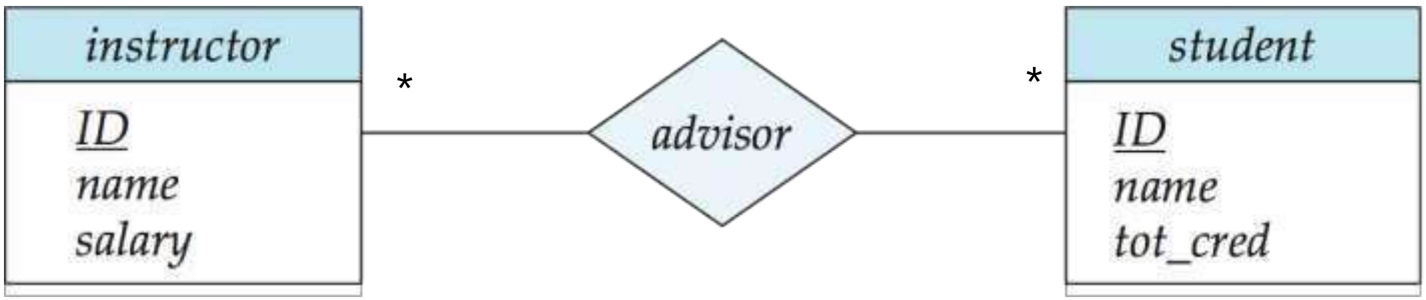
- One-to-one relationship:
  - A student is associated with at most one *instructor* via the relationship *advisor*
  - A *student* is associated with at most one *department* via *stud_dept*

ONE-TO-MANY RELATIONSHIP

- one-to-many relationship between an *instructor* and a *student*
  - an instructor is associated with several (including 0) students via *advisor*
  - a student is associated with at most one instructor via advisor,

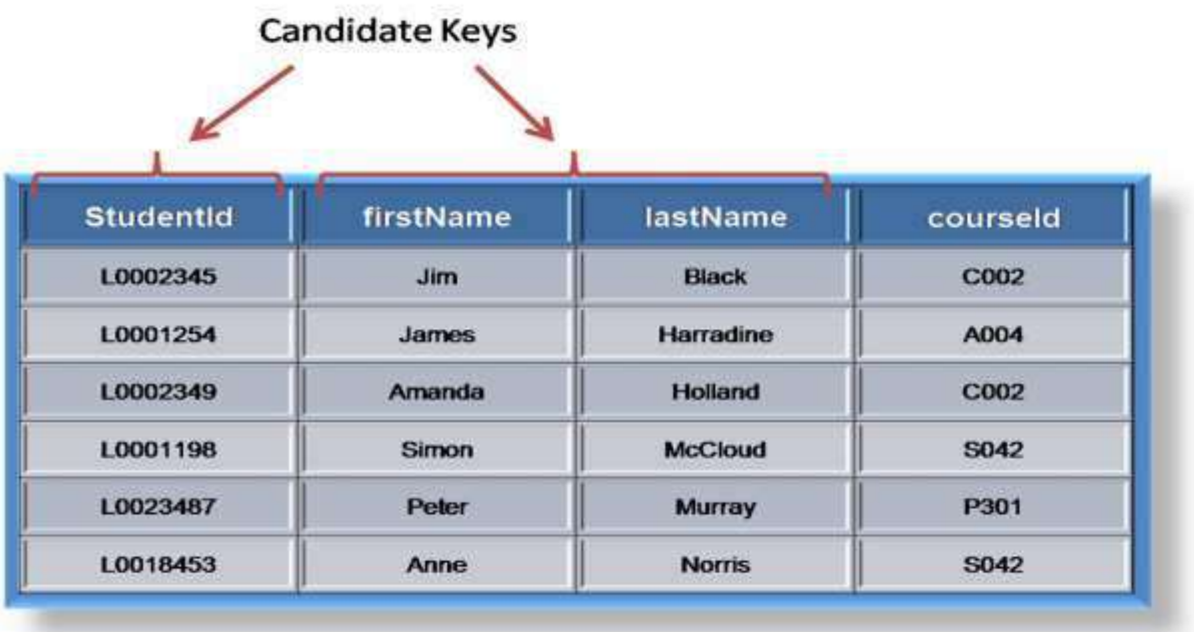
Many-to-Many Relationship

- An instructor is associated with several (possibly 0) students via *advisor*
- A student is associated with several (possibly 0) instructors via *advisor*



Different Types of Keys

- A **candidate key** of an entity set is a minimal super key
  - *ID* is candidate key of *instructor*
  - *course_id* is candidate key of *course*



StudentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

PRIMARY KEY

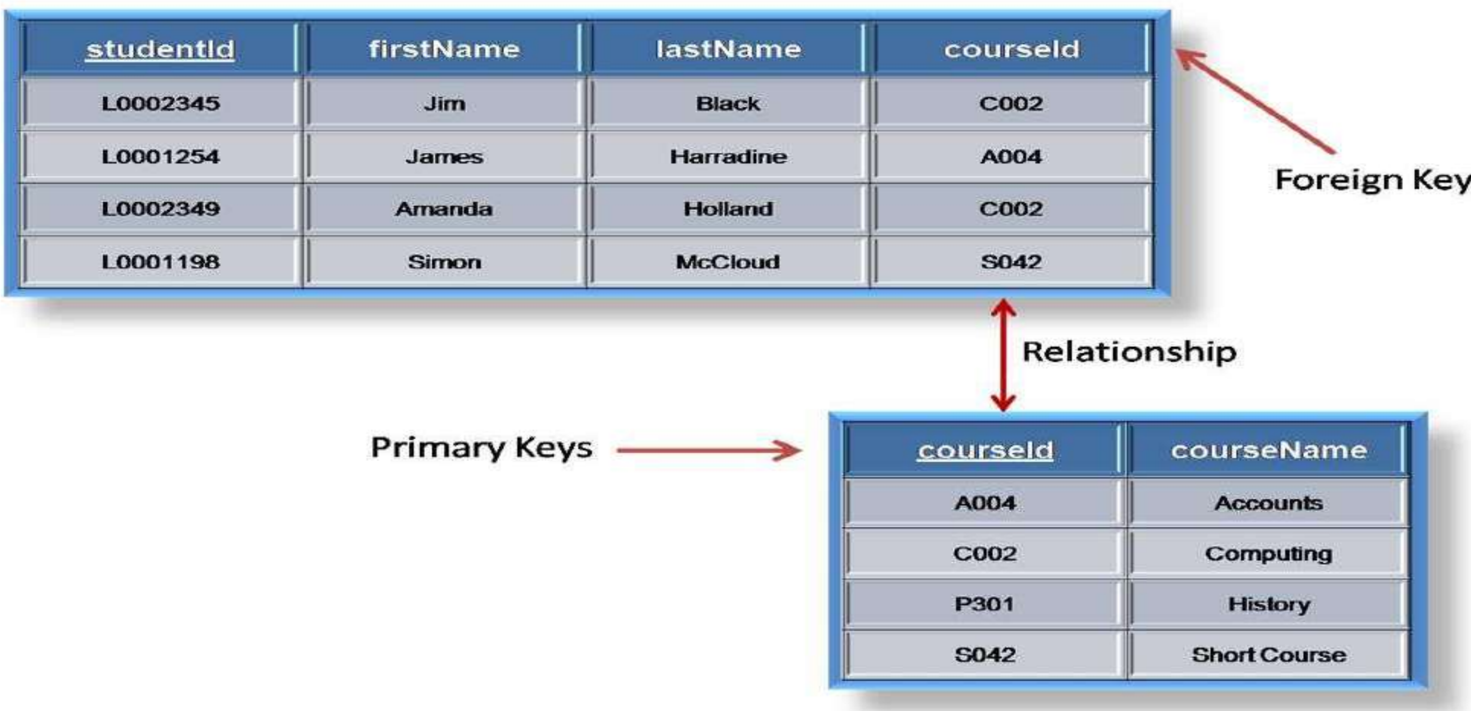
- A primary key is a candidate key that is most appropriate to be the main reference key for the table. As its name suggests, it is the primary key of reference for the table and is used throughout the database to help establish relationships with other tables.
- **The primary key must contain unique values, must never be null and uniquely identify each record in the table**

Primary Keys

StudentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

FOREIGN KEY

- A foreign key is generally a primary key from one table that appears as a field in another where the first table has a relationship to the second. In other words, if we had a table A with a primary key X that linked to a table B where X was a field in B, then X would be a foreign key in B



studentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042

courseId	courseName
A004	Accounts
C002	Computing
P301	History
S042	Short Course

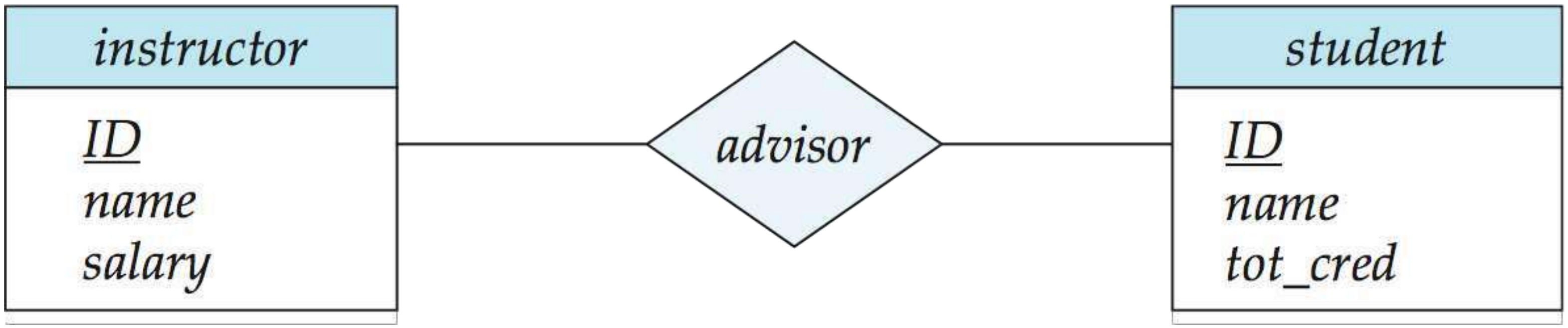
DIFFERENT TYPES OF KEYS

A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.

Example:

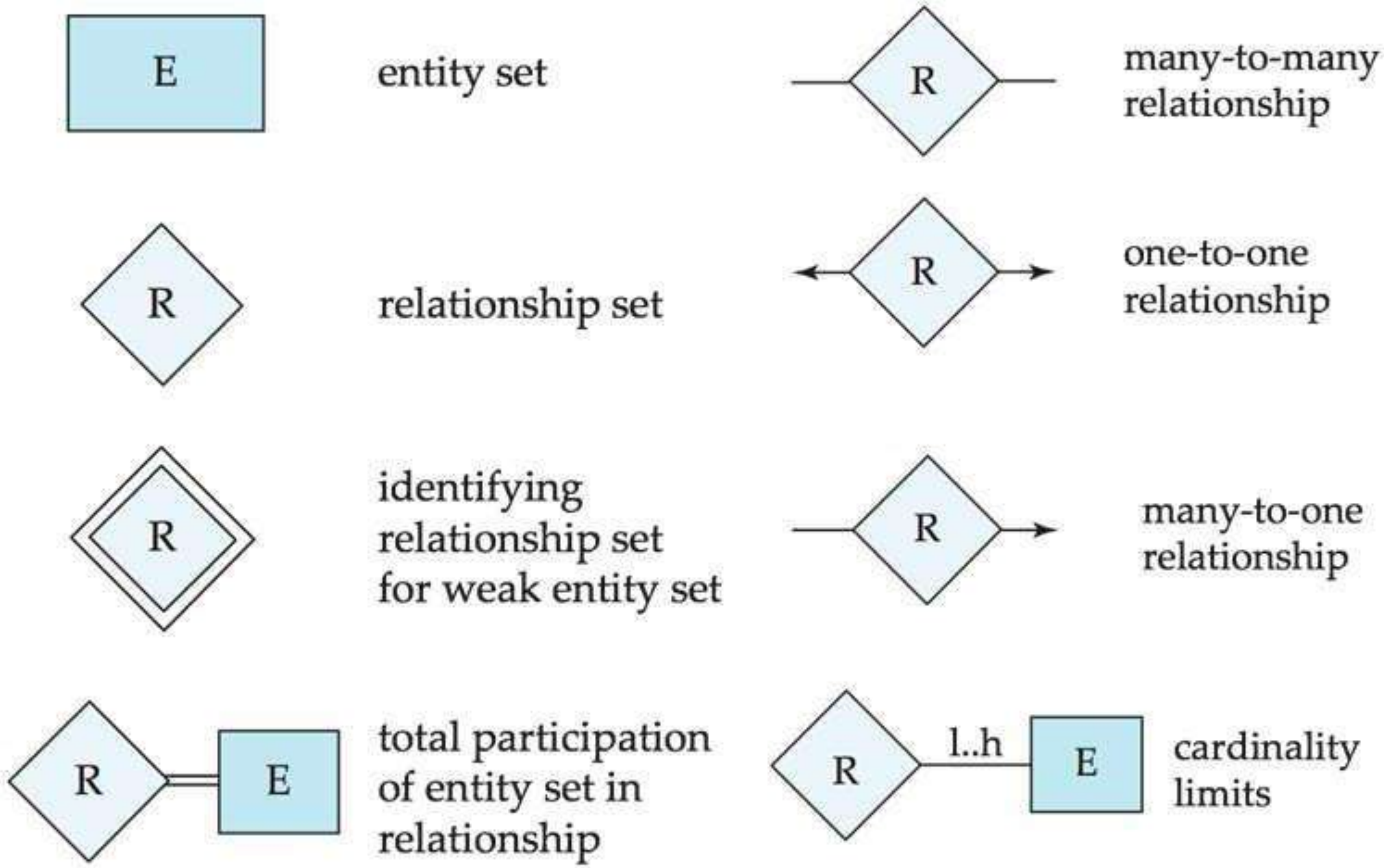
- {Student ID,FirstName }
- {Student ID, LastName }
- {Student ID,FirstName,LastName}

E-R DIAGRAMS



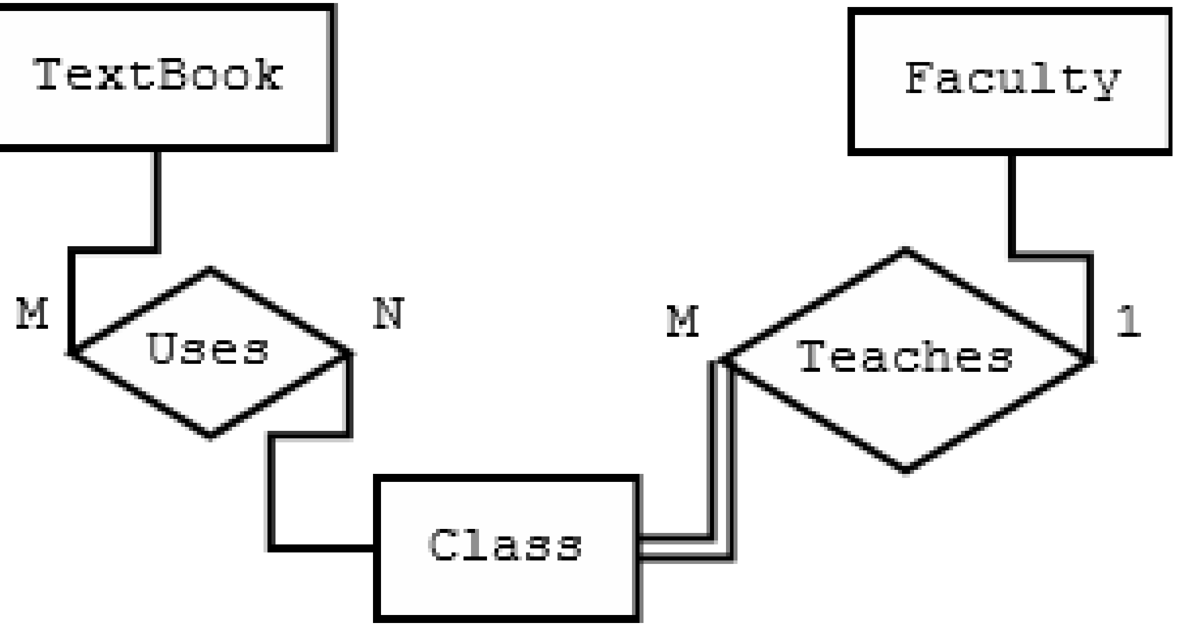
- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Attributes listed inside entity rectangle. Or , as oval shape along with the rectangle.
- Underline indicates primary key attributes

SUMMARY OF SYMBOLS USED IN E-R NOTATION

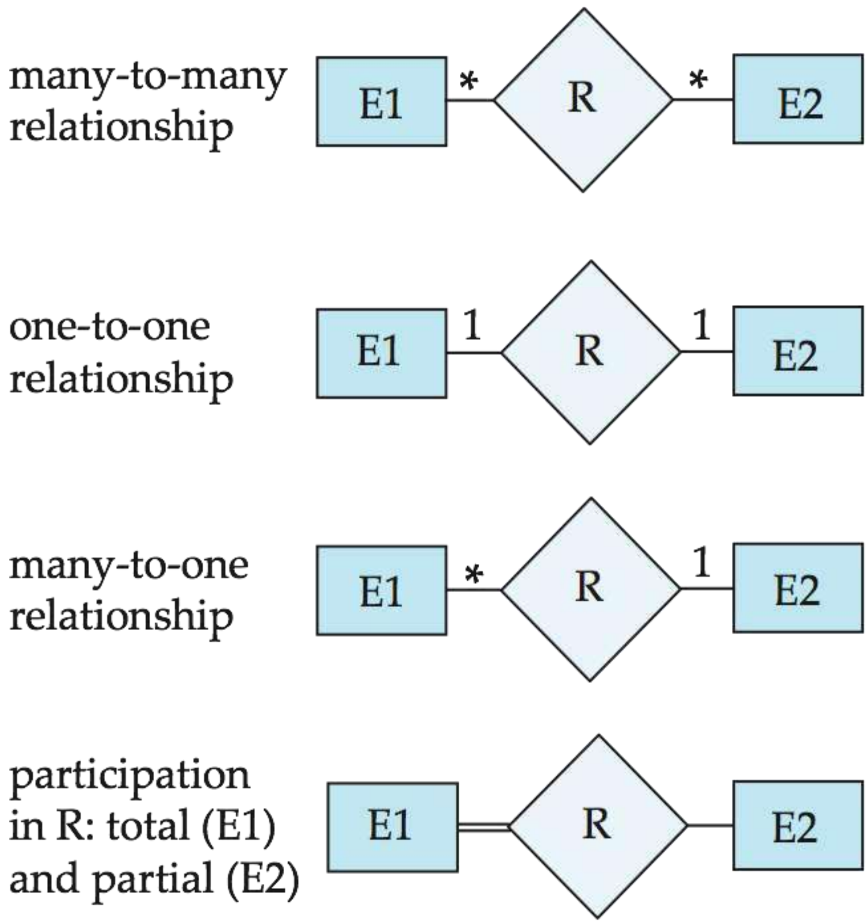


TOTAL PARTICIPATION OF ENTITY SET

- E.g., A *Class* entity cannot exist unless related to a *Faculty* member entity

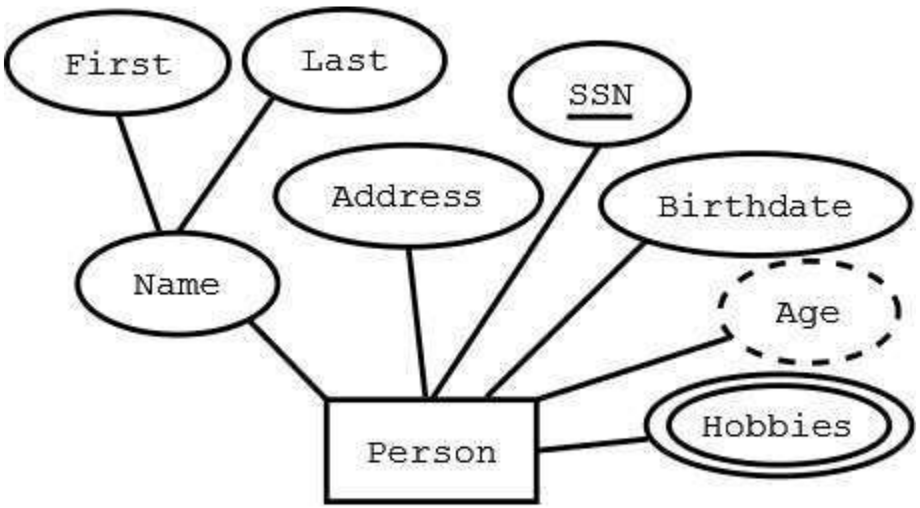


SUMMARY OF SYMBOLS USED IN E-R NOTATION



Summary of Symbols Used in E-R Notation

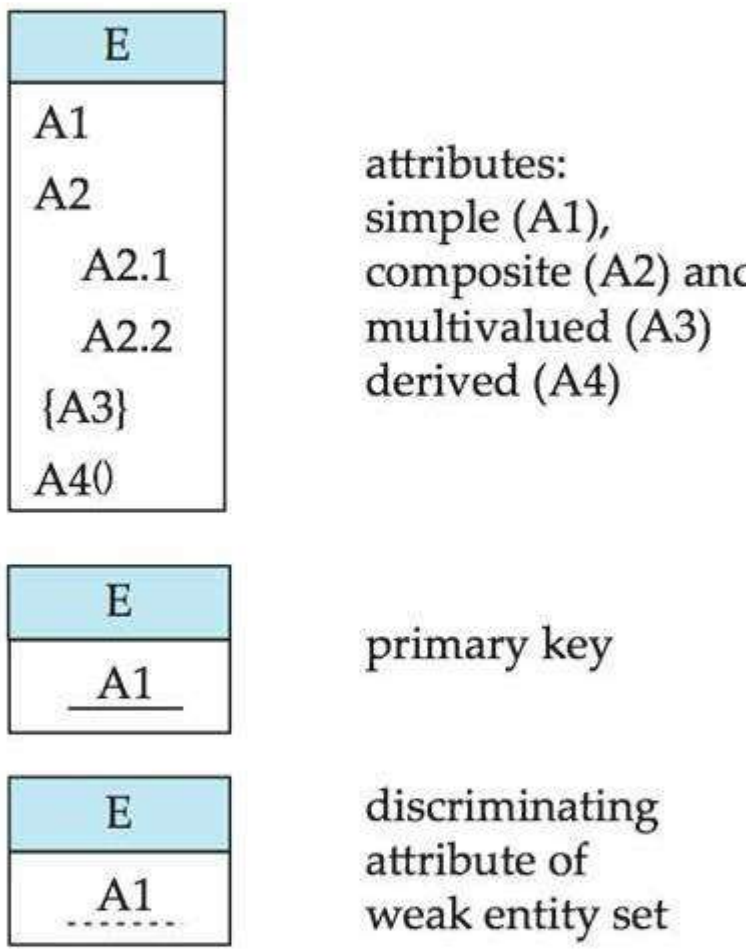
Representing attributes



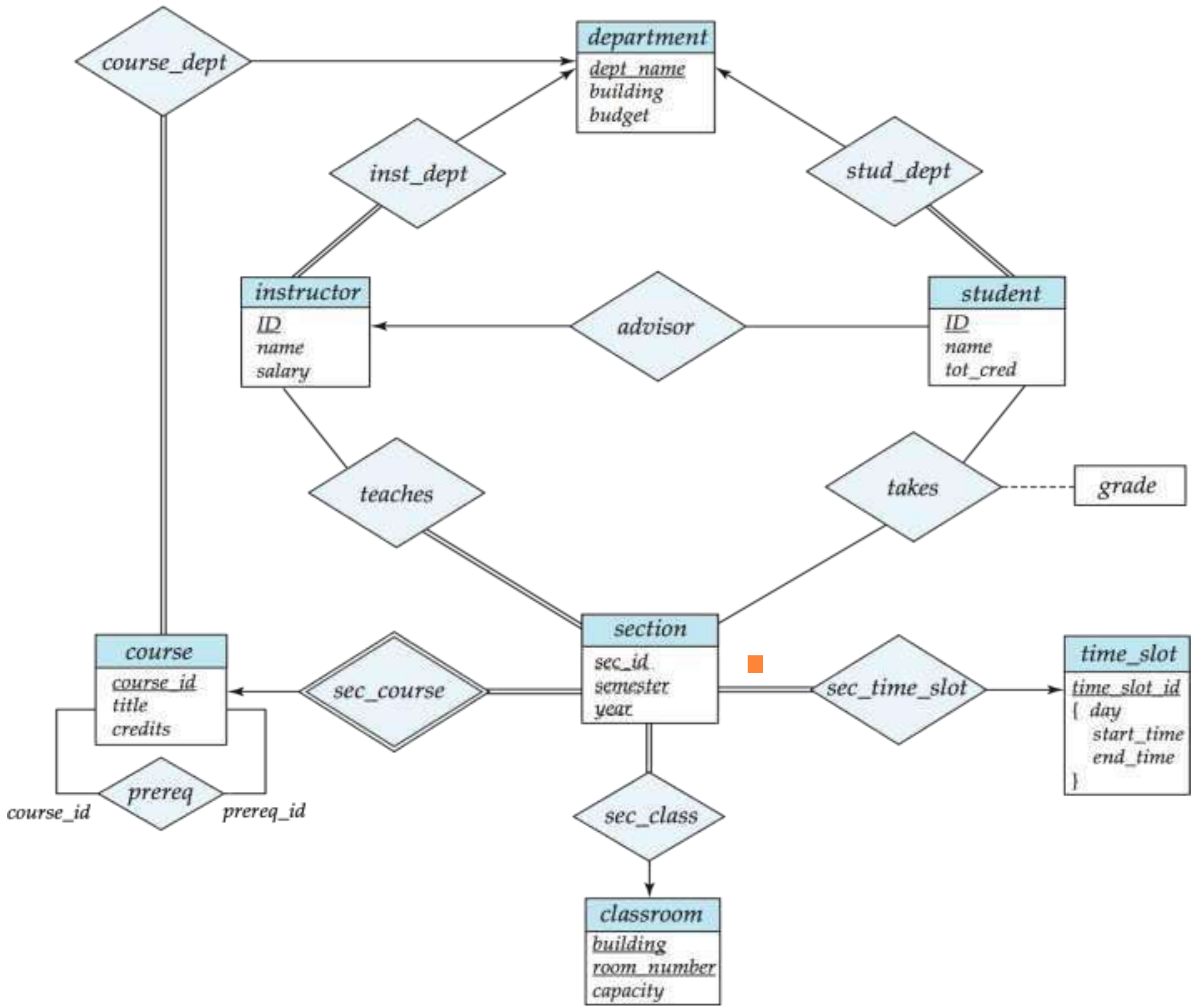
- **Rectangle** -- Entity
- **Ellipses** -- Attribute (<u>underlined</u> attributes are [part of] the primary key)
- **Double ellipses** -- multi-valued attribute
- **Dashed ellipses**-- derived attribute, e.g. age is derivable from birthdate and current date.

Summary of Symbols Used in E-R Notation

- Representing attributes



E-R Diagram for a University



Invitation to Computer Science 5th Edition

Chapter 15

Artificial Intelligence

Objectives

In this chapter, you will learn about:

- A division of labor
- Recognition tasks
- Reasoning tasks
- Robotics

Introduction

- Artificial intelligence (AI)
  - Explores techniques for incorporating aspects of intelligence into computer systems
- Turing test
  - Allows a human to interrogate two entities, both hidden from the interrogator

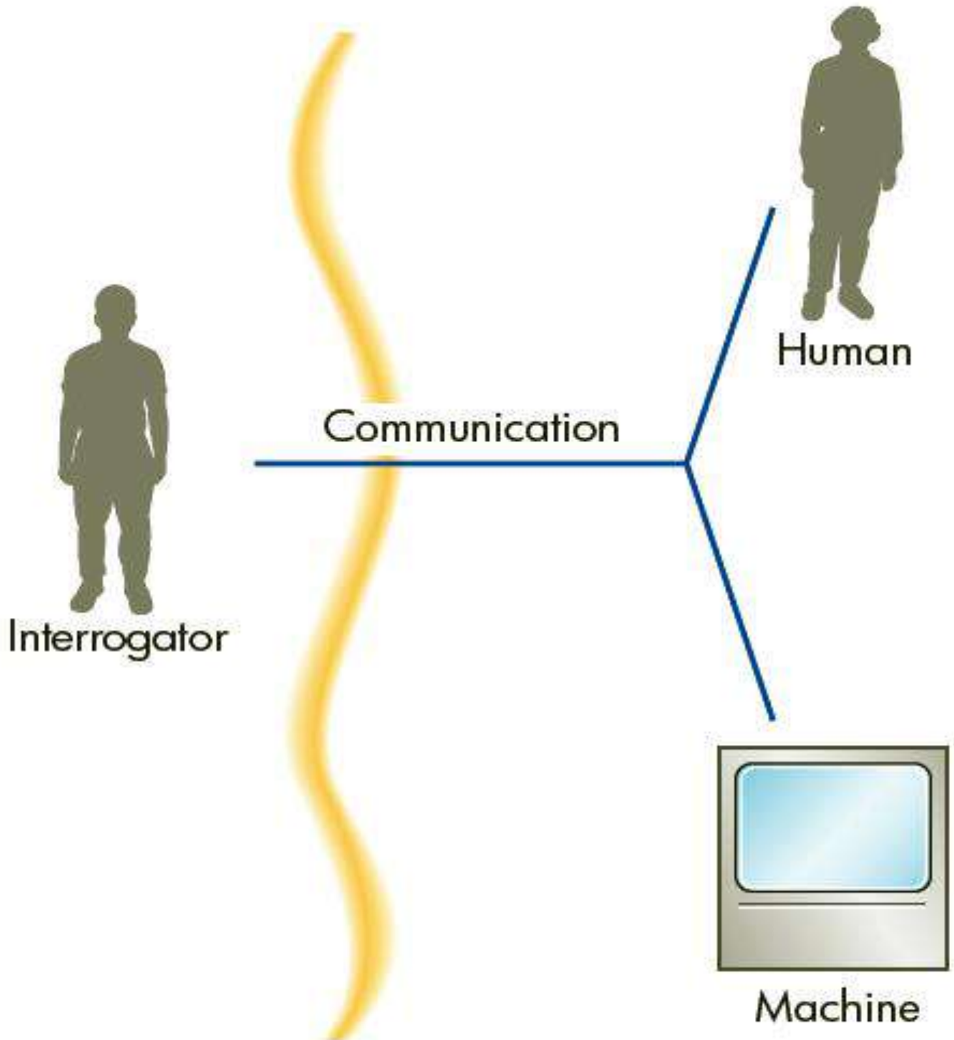


Figure 15.1 The Turing Test

A Division of Labor

- Computational tasks
  - Adding a column of numbers
  - Sorting a list of numbers into numerical order
  - Searching for a given name in a list of names
  - Managing a payroll
  - Calculating trajectory adjustments for the space shuttle

A Division of Labor (continued)

- Recognition tasks
  - Recognizing your best friend
  - Understanding the spoken word
  - Finding the tennis ball in the grass in your backyard

A Division of Labor (continued)

- Reasoning tasks
  - Planning what to wear today
  - Deciding on the strategic direction a company should follow for the next five years
  - Running the triage center in a hospital emergency room after an earthquake

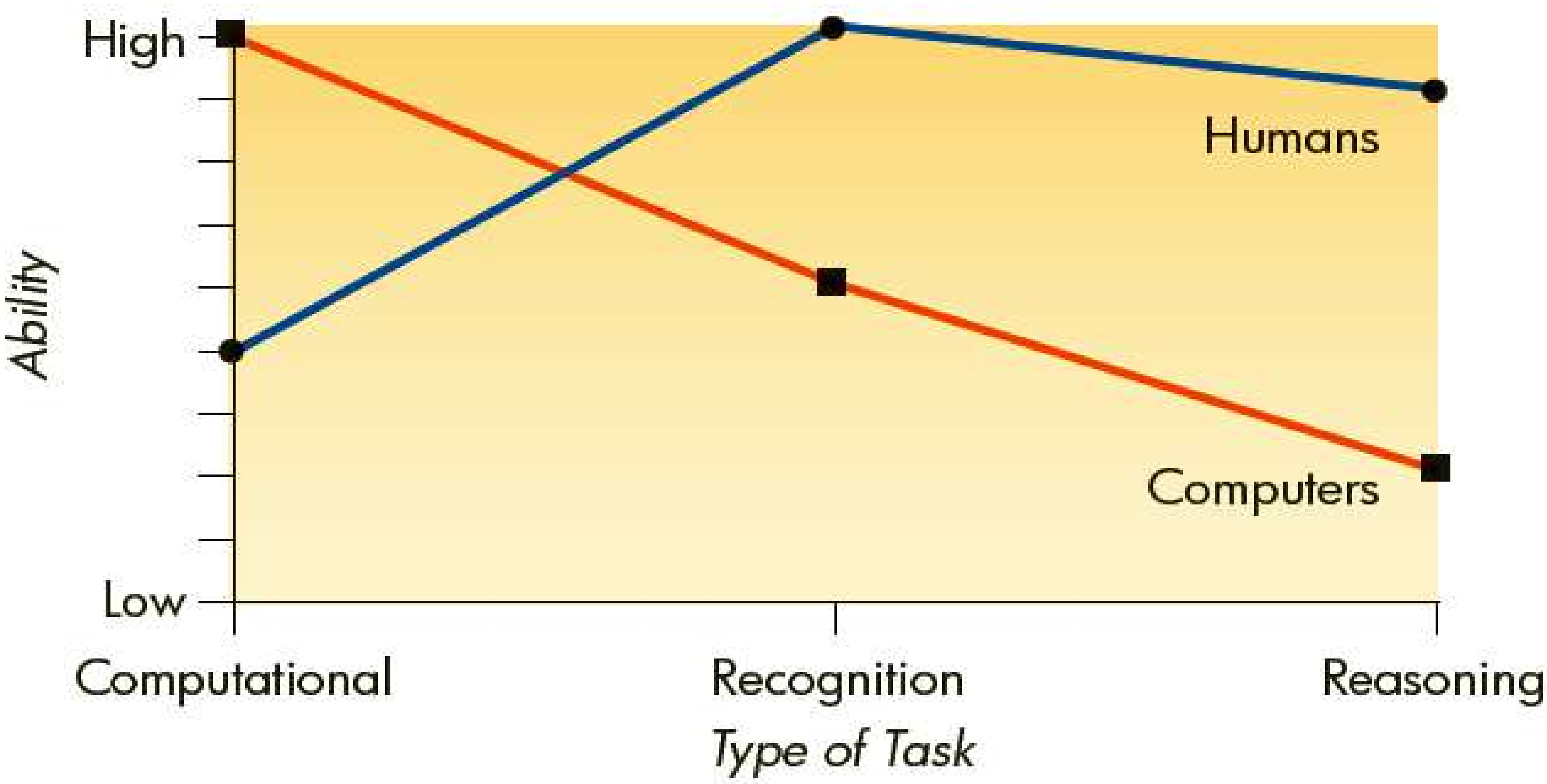


Figure 15.2 Human and Computer Capabilities

Recognition Tasks

- Neuron
  - Cell capable of receiving stimuli, in the form of electrochemical signals, from other neurons through its many **dendrites**
  - Can send stimuli to other neurons through its single **axon**
- Artificial neural networks
  - Can be created by simulating individual neurons in hardware and connecting them in a massively parallel network of simple devices

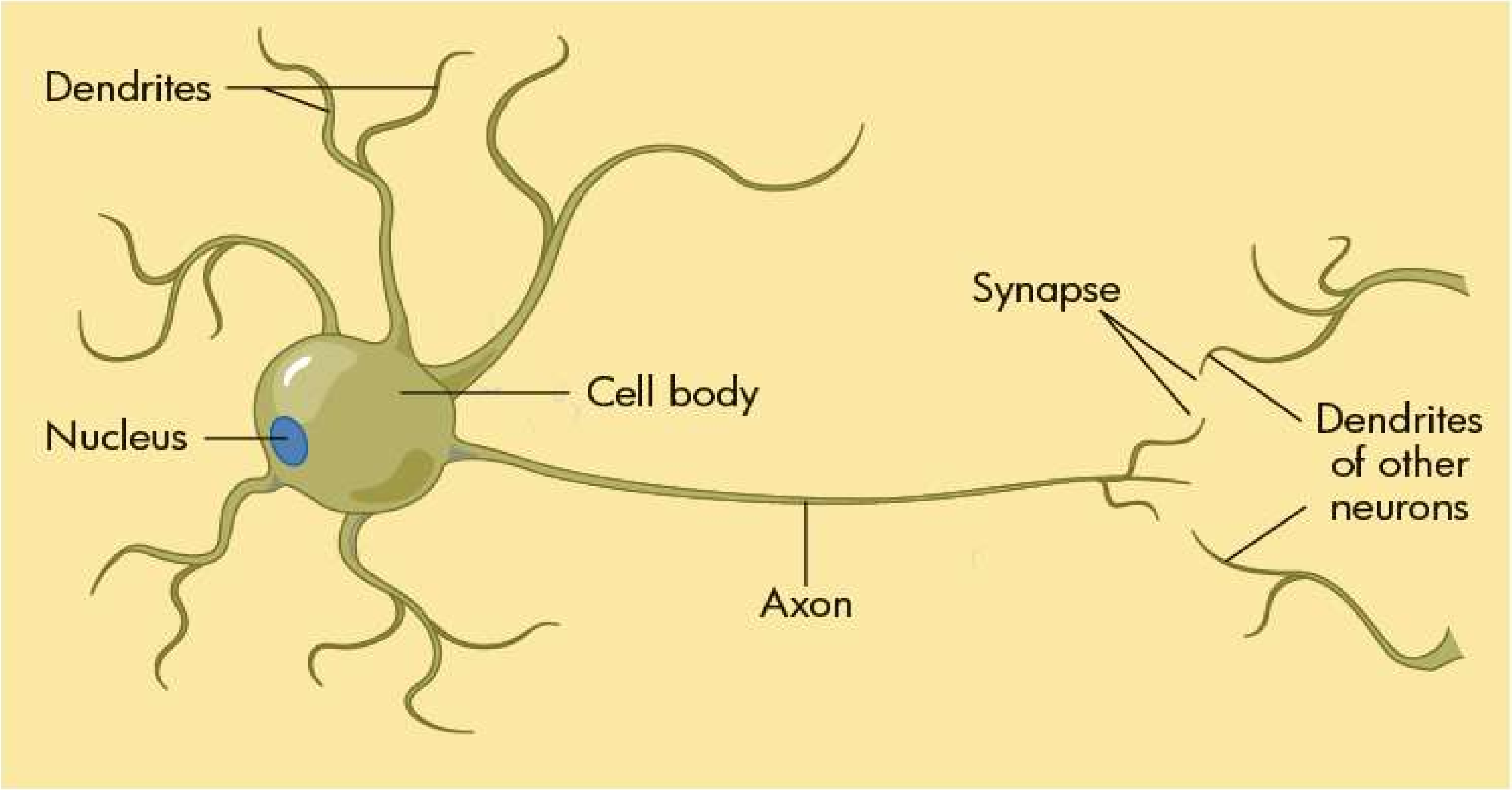


Figure 15.4 A Neuron

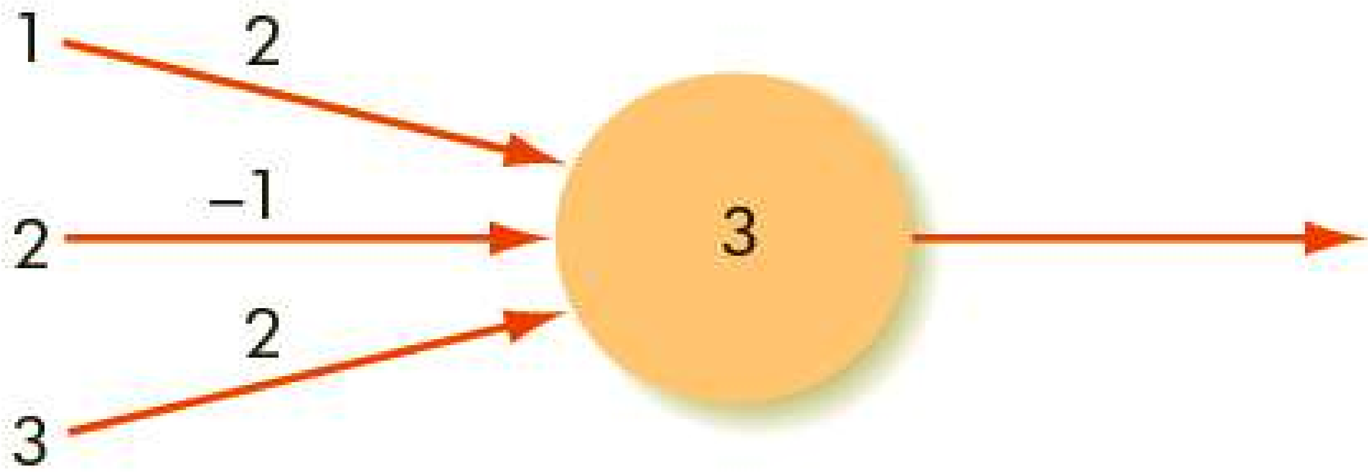


Figure 15.5 One Neuron with Three Inputs

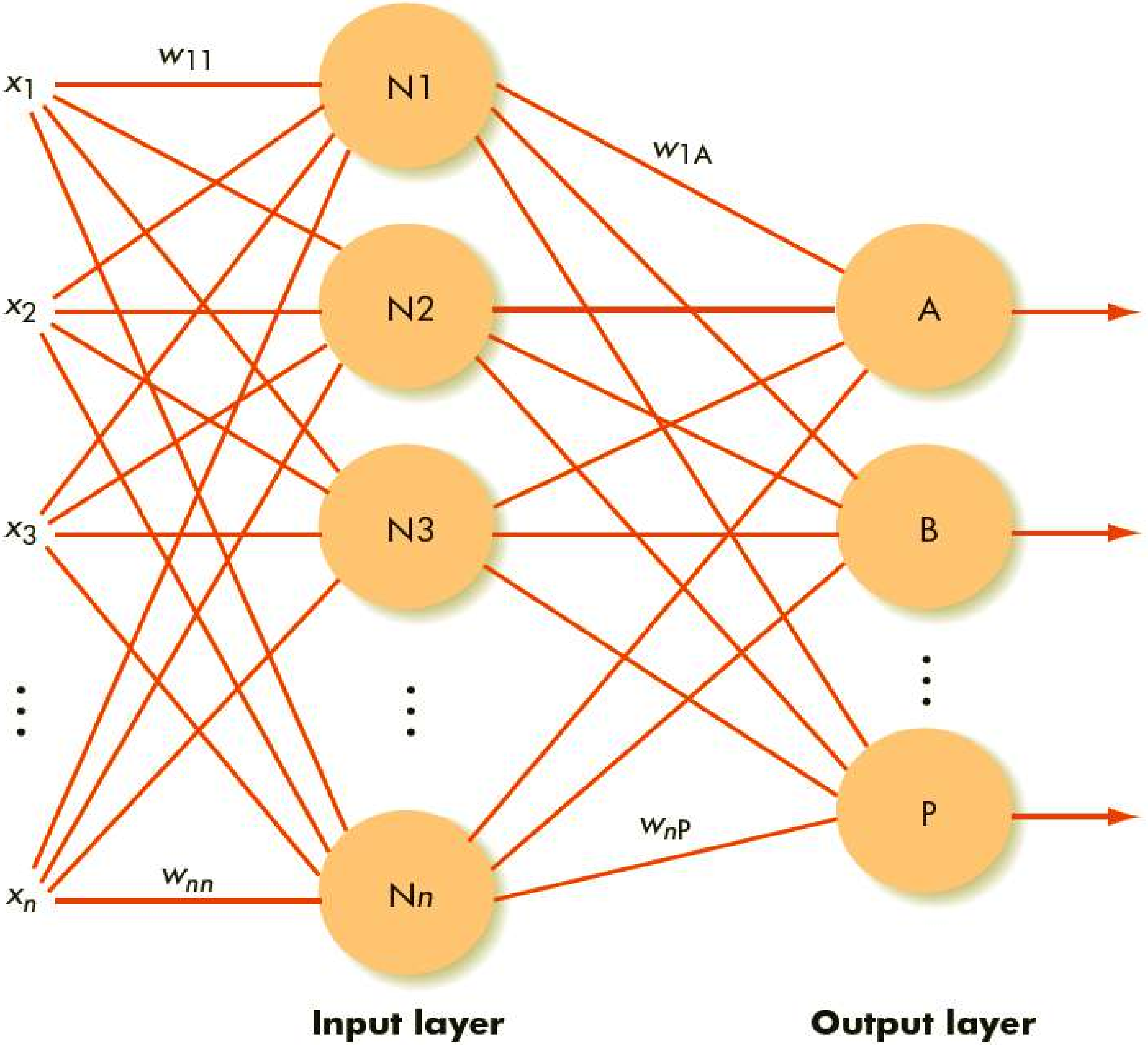


Figure 15.6 Neural Network Model

Recognition Tasks (continued)

- Neural network
  - Can learn from experience by modifying the weights on its connections
  - Can be given an initial set of weights and thresholds that is simply a first guess
  - Network is then presented with **training data**
- Back propagation algorithm
  - Eventually causes the network to settle into a stable state where it can correctly respond to all inputs in the training set

Reasoning Tasks

- Characteristic of human reasoning
  - Ability to draw on a large body of facts and past experience to come to a conclusion
- Artificial intelligence specialists try to get computers to emulate this characteristic

Intelligent Searching

- Decision tree for a search algorithm
  - Illustrates the possible next choices of items to search if the current item is not the target
- Decision tree for sequential search is linear
- Classical search problem benefits from two simplifications
  - Search domain is a linear list
  - We seek a perfect match

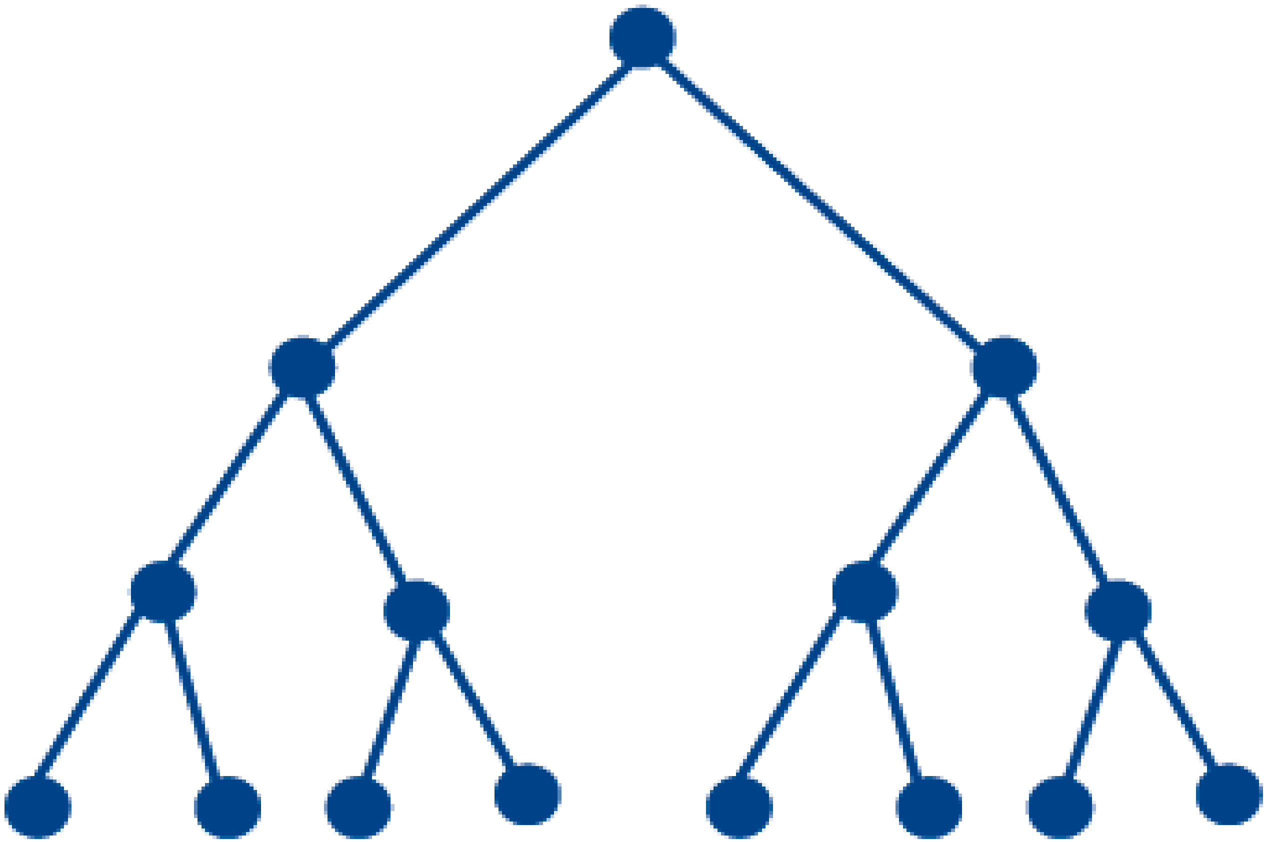


Figure 15.11 Decision Tree for Binary Search

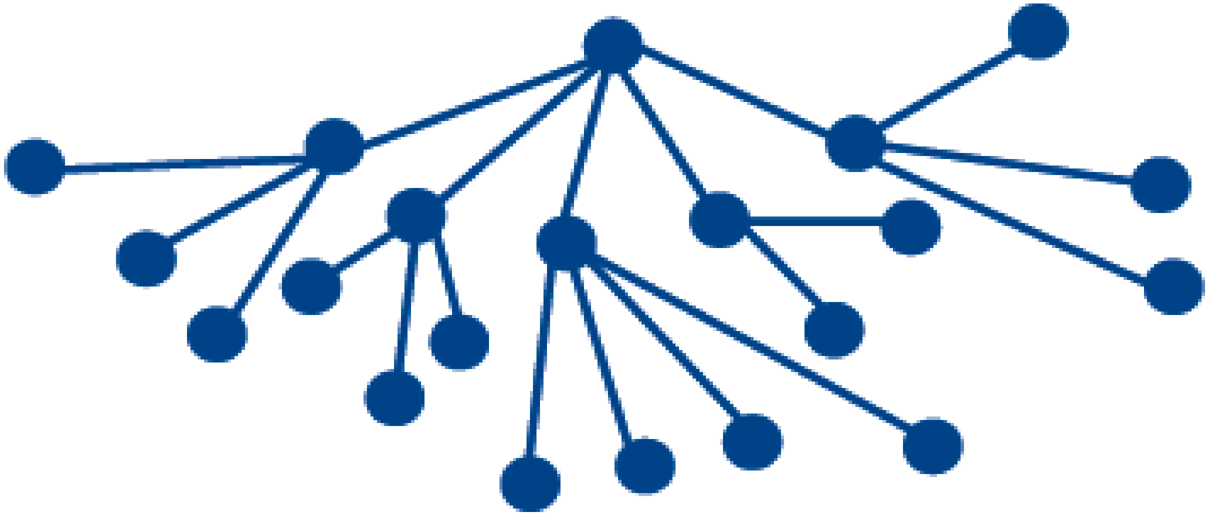
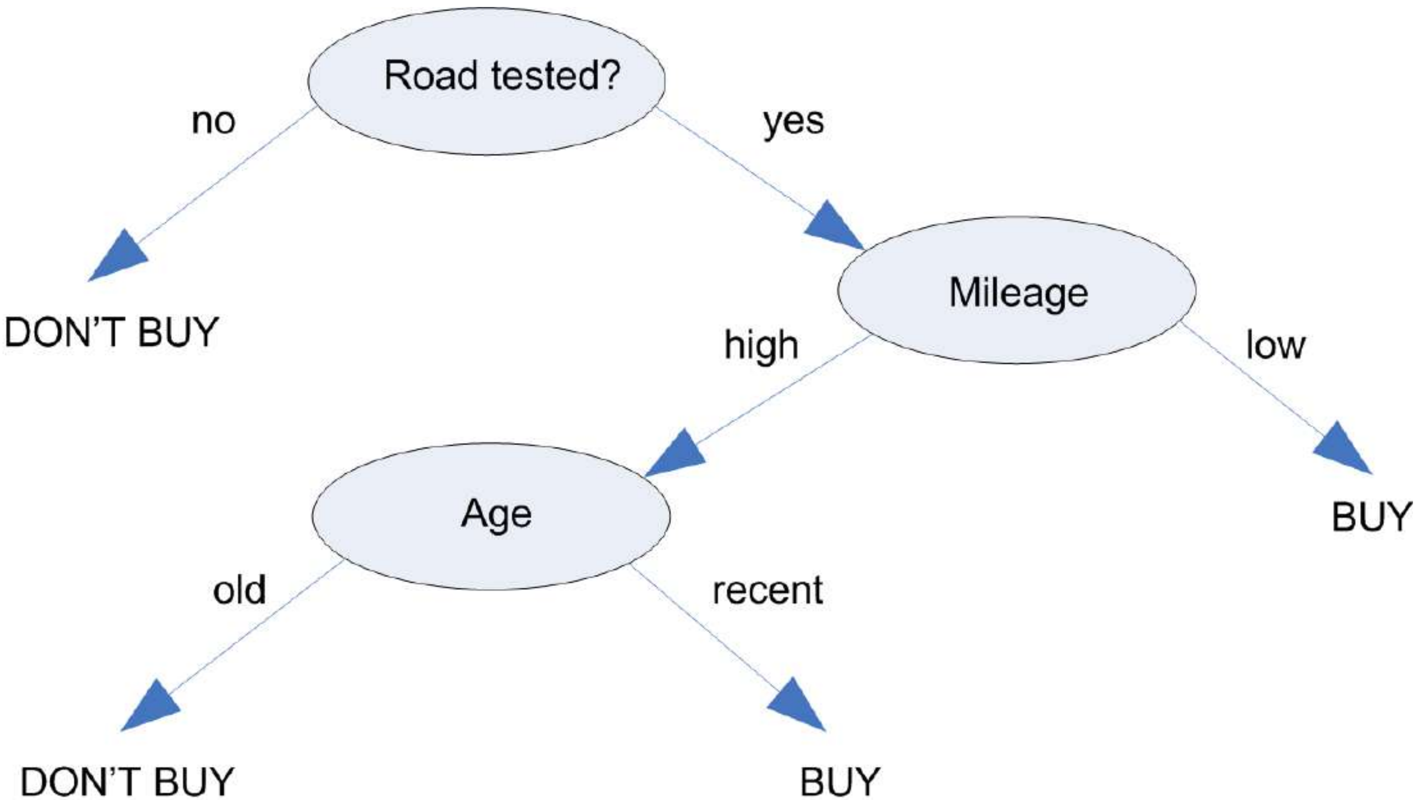
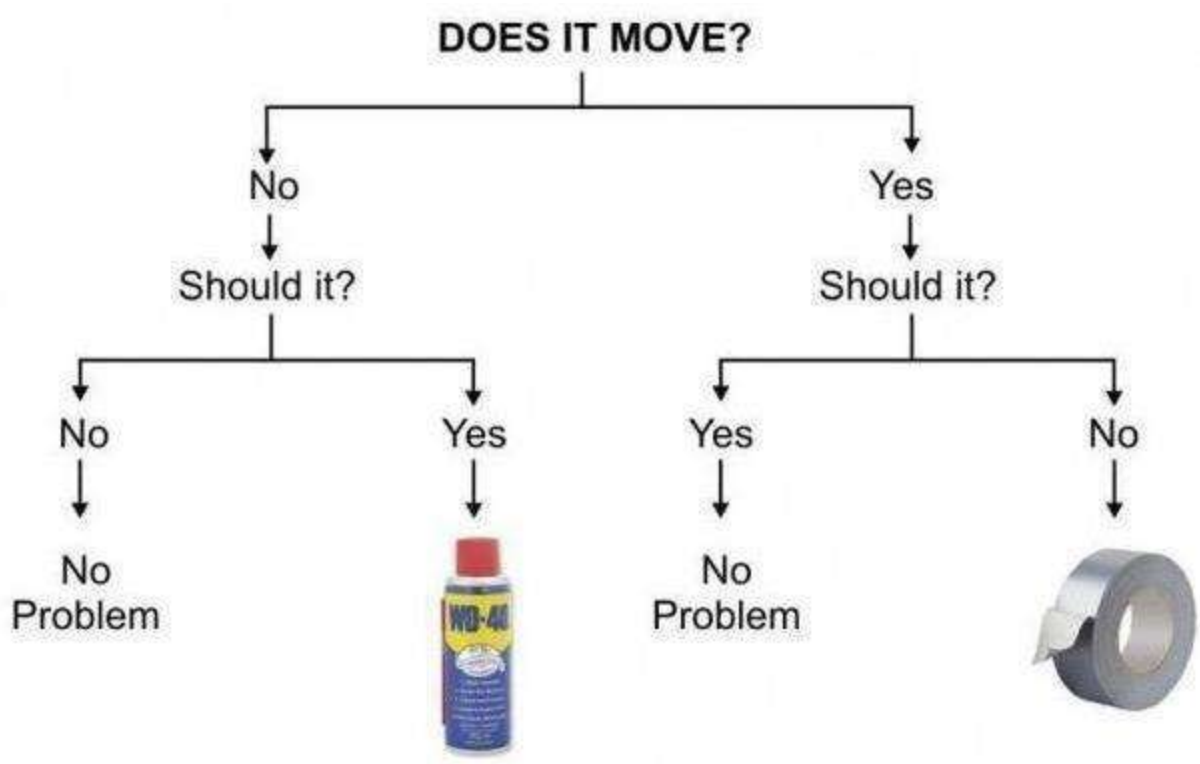


Figure 15.12 A State-Space Graph with Exponential Growth

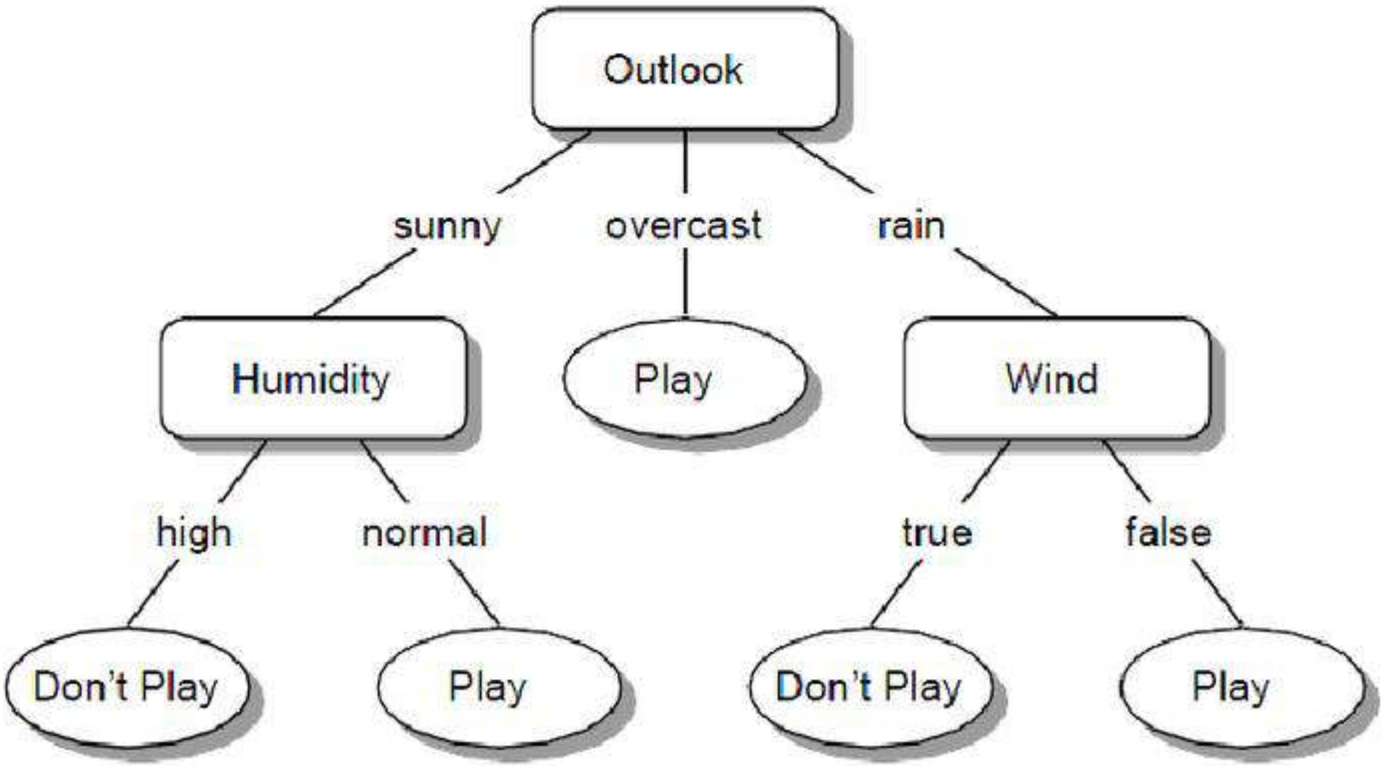
Decision Tree Example



Decision Tree Example



Decision Tree Example



Classification: Definition

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.

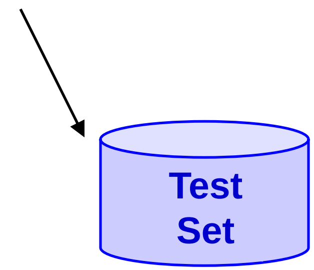
- Goal: <u>previously unseen</u> records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

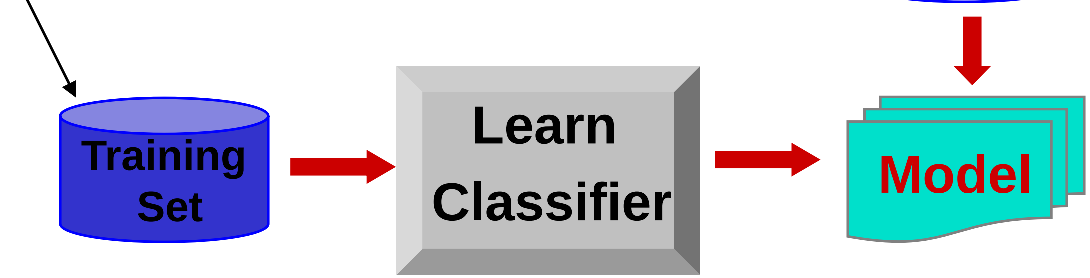
Classification Example

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Refund	Marital Status	Taxable Income	Cheat
No	Single	75K	?
Yes	Married	50K	?
No	Married	150K	?
Yes	Divorced	90K	?
No	Single	40K	?
No	Married	80K	?





Classification: Application 1

- Ad Click Prediction
  - Goal: Predict if a user that visits a web page will click on a displayed ad. Use it to target users with high click probability.
  - Approach:
    - Collect data for users over a period of time and record who clicks and who does not. The {click, no click} information forms the class attribute.
    - Use the history of the user (web pages browsed, queries issued) as the features.
    - Learn a classifier model and test on new users.

Classification: Application 2

- Fraud Detection
  - Goal: Predict fraudulent cases in credit card transactions.
  - Approach:
    - Use credit card transactions and the information on its account-holder as attributes.
      - When does a customer buy, what does he buy, how often he pays on time, etc
    - Label past transactions as fraud or fair transactions. This forms the class attribute.
    - Learn a model for the class of the transactions.
    - Use this model to detect fraud by observing credit card transactions on an account.

Clustering Definition

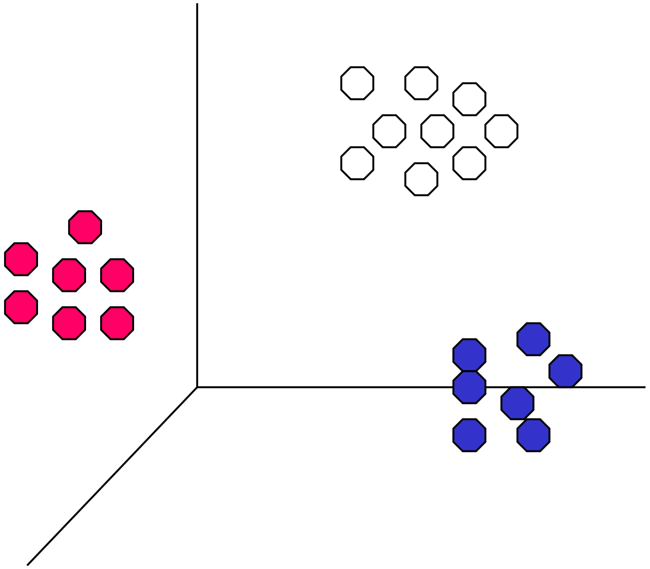
- Given a set of data points, each having a set of attributes, and a similarity measure among them, find clusters such that
  - Data points in one cluster are more similar to one another.
  - Data points in separate clusters are less similar to one another.
- Similarity Measures?
  - Euclidean Distance if attributes are continuous.
  - Other Problem-specific Measures.

Illustrating Clustering

Euclidean Distance Based Clustering in 3-D space.

Intracluster distances are minimized

Intercluster distances are maximized



Clustering: Applications

- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- Land use: Identification of areas of similar land use in an earth observation database
- Insurance: Identifying groups of motor insurance policy holders with a high average claim cost
- City-planning: Identifying groups of houses according to their house type, value, and geographical location
- Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults

Expert Systems

- Rule-based system
  - Attempts to mimic the human ability to engage pertinent facts and string them together in a logical fashion to reach some conclusion
  - Must contain these two components
    - A knowledge base
    - An inference engine

Robotics

- Uses for robots in manufacturing, science, the military, and medicine
  - Assembling automobile parts
  - Packaging food and drugs
  - Placing and soldering wires in circuits
  - Bomb disposal
  - Welding
  - Radiation and chemical spill detection

Robotics (continued)

- Two strategies characterize robotics research
  - **Deliberative strategy**: says that the robot must have an internal representation of its environment
  - **Reactive strategy**: uses heuristic algorithms to allow the robot to respond directly to stimuli from its environment

Summary

- Artificial intelligence
  - Explores techniques that incorporate aspects of intelligence into computer systems
- Categories of tasks
  - Computational, recognition, and reasoning
- Neural networks
  - Simulate individual neurons in hardware and connect them in a massively parallel network

Summary (continued)

- Intelligent agent interacts with a user
- Rule-based systems
  - Attempt to mimic the human ability to engage pertinent facts and combine them in a logical way to reach some conclusion
- Robots can perform many useful tasks

COMPUTER ETHICS

Slide Credits to: K. Avinash, P. Dharun, M. Hariprasadh,
C.K. Jaganathan, S.Jishu
Dr. Ahmet Kılıç

INTRODUCTION

- Ethics is a set of moral principles that govern the behavior of a group or individual.
- likewise, computer ethics is set of moral principles that regulate the useof computers.

Common issues of computer ethics

Some common issues of computer ethics include intellectual property rights such as copyrighted electronic content, privacy concerns, and how computers affect society.

Contd….

For example, while it is easy to duplicate copyrighted electronic or digital content, computer ethics would suggest that it is wrong to do so without the author's approval.

And while it may be possible to access someone's personal information on a computer system, computer ethics would advise that such an action is unethical.

INTELLECTUAL

You have certainly heard the word property before: it is generally used to mean a possession, or more specifically, something to which theowner has legal rights.

You might have also encountered the phrase intellectual property. This term has become more commonplace during the past few years, especially in the context of computer ethics. But what exactly does it refer to?

Contd…

Intellectual property refers to creations of the intellect (hence, the name): inventions, literary and artistic works, symbols, names, images, and designs used in commerce are a part of it.

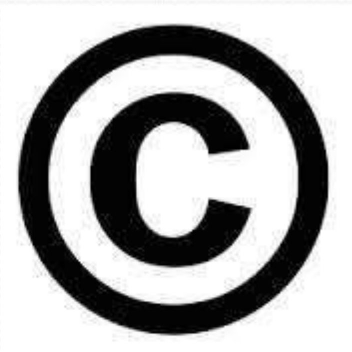
Intellectual property is usually divided into two branches, namely *industrial property* which broadly speaking protects inventions and *copyright*, which protects literary and artistic works.

CATEGORISING INTELLECTUAL PROPERY

- Intellectual property is divided into twocategories:
- *Industrial property*, which includes inventions (patents), trademarks, industrial designs, commercial names, designations and geographic indications (location specific brands) etc.
- *Copyright*, which includes literary and artistic works such as novels, poems and plays, films, musical works, artistic works such as drawings, paintings, photographs, sculptures, and architectural designs.

Copy rights

Copyright is a legal concept, enacted by most governments, giving the creator of an original work exclusive rights to it, usually for a limited time.



WHAT IT CAN PROTECT AND WHAT NOT

In summary, copyright laws protect intellectual property which includes literary and artistic works such as novels, poems and plays, films, musical works, artistic works such as drawings, paintings, photographs and sculptures, and architectural designs.

But unlike protection of inventions, copyright law protects only the form of expressions of ideas, not the ideas themselves.

Remember that a created work is considered protected as soon as it exists, and a public register of copyright protected work is not necessary.

COPY RIGHT ON INTERNET

- But what of works made available to the public on the Internet? Are they at all protected by copyright? Once again, yes! For works made available over a communications network (such as the Internet), the copyright protects original authorship.
- But, according to the Copyright Law, it does not protect ideas, procedures, systems, or methods of operation. This means that once such an online work has been made public, nothing in the copyright laws prevents others from developing another work based on similar principles, or ideas.

REAL PEPOLE EXIST BEHIND THE COMPUTERS

You are dealing with people, not machines. So think twice before you click on Send button in the mail/chat window

You are not the only one using the network

Keep these other people in mind when you say something on a network.

PROTECT YOUR PRIVACY

- Just as you would in the real world, be aware of risks, fraud and false information which exists on the Internet. Use common sense when deciding whether information is valid. Don't trust or spread further any information about which you are in doubt. Always try to obtain reliable information.
- Protect your personal information to keep someone from using it in an unethical way. (For example, when you enter a prize contest, your name, address, and phone number may be given to a dealer of personal information.)

AVOID SPAMMING

- Spamming is sending unsolicited bulk and/or commercial messages over the Internet.
- Spamming is morally bad if it is intended to destroy and done by infringing on the right of privacy of others.
- It could be good if the message sent benefits the recipients, like giving out warnings or useful information to others.

SOFTWARE PRIVACY

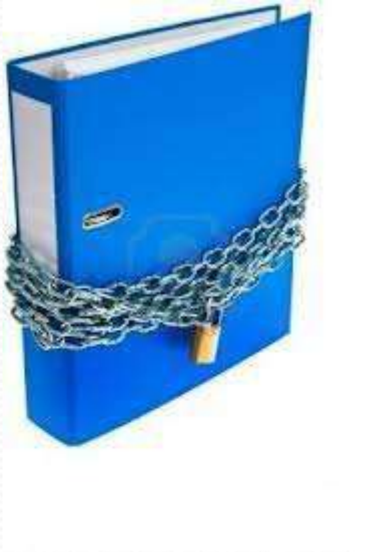
- Software piracy is morally bad when someone reproduces a copy of the software and sells it for profit, produces exactly the same or similar version without giving proper credit to the original author, or simply produces it and distributes it to others.
- It is not immoral to copy the software if someone who has a licensed copy of the software and simply makes a backup copy of the original. One back-up copy of the commercial software can be made, but the back-up copy cannot be used except when the original package fails or is destroyed.

PLAGIARISM

- Plagiarism is copying someone else's work and then passing it off as one's own. It is morally bad because it is an act of stealing.
- Copying programs written by other programmers and claiming it as your own could be an act of plagiarism. It involves lying, cheating, theft, and dishonesty.

FILE PRIVACY

Any computer document produced either by an individual in his private home or in his office should remain private. No one has should open any document unless authorized by the individual who created the file himself.



TCK Bilişim Suçları

5237 sayılı TCK’ unda bilişim suçları; “Bilişim alanında suçlar” bölümünde düzenlenmekle birlikte, ayrıca çeşitli bölümlerde de bilişim sistemlerinin kullanılması suretiyle işlenmesi mümkün olan suç tiplerine yer verilmiştir. "Bilişim alanında suçlar" bölümünde yer alan;

a)TCK 243. maddesinde “Bilişim sistemine girme”,
b)TCK 244. maddesinde “Sistemi engelleme, bozma, verileri yok etme veya değiştirme”,
c)TCK 245. maddesinde “Banka veya kredi kartlarının kötüye kullanılması ”
d)TCK 245/A maddesinde ''Yasak Cihaz veya Programlar'' suçları düzenlenmiştir. Bu ek madde 6698 sayılı yasa ile 07/04/2016 tarihinde 29677 sayılı Resmi gazete de yayınlanarak yürürlüğe girmiştir.

BİLİŞİM SİSTEMİNE GİRME
TCK MADDE 243

(1) Bir bilişim sisteminin bütününe veya bir kısmına, hukuka aykırı olarak giren *veya* orada kalmaya devam eden kimseye bir yıla kadar hapis veya adlî para cezası verilir.
(2) Yukarıdaki fıkrada tanımlanan fiillerin bedeli karşılığı yararlanılabilen sistemler hakkında işlenmesi hâlinde, verilecek ceza yarı oranına kadar indirilir.
(3) Bu fiil nedeniyle sistemin içerdiği veriler yok olur veya değişirse, altı aydan iki yıla kadar hapis cezasına hükmolunur.
(4) (Ek fıkra: 24/03/2016-6698 S.K./30. md) Bir bilişim sisteminin kendi içinde veya bilişim sistemleri arasında gerçekleşen veri nakillerini, sisteme girmeksizin teknik araçlarla hukuka aykırı olarak izleyen kişi, bir yıldan üç yıla kadar hapis cezası ile cezalandırılır.

SİSTEMİ ENGELLEME, BOZMA, VERİLERİ YOK ETME VEYA DEĞİŞTİRME
TCK MADDE 244

[1] Bir bilişim sisteminin işleyişini engelleyen veya bozan kişi, bir yıldan beş yıla kadar hapis cezası ile cezalandırılır.
[2] Bir bilişim sistemindeki verileri bozan, yok eden, değiştiren veya erişilmez kılan, sisteme veri yerleştiren, var olan verileri başka bir yere gönderen kişi, altı aydan üç yıla kadar hapis cezası ile cezalandırılır.
[3] Bu fiillerin bir banka veya kredi kurumuna ya da bir kamu kurum veya kuruluşuna ait bilişim sistemi üzerinde işlenmesi halinde, verilecek ceza yarı oranında artırılır.
[4] Yukarıdaki fıkralarda tanımlanan fiillerin işlenmesi suretiyle kişinin kendisinin veya başkasının yararına haksız bir çıkar sağlamasının başka bir suç oluşturmaması hâlinde, iki yıldan altı yıla kadar hapis ve beşbin güne kadar adlî para cezasına hükmolunur.

BANKA VEYA KREDİ KARTLARININ KÖTÜYE KULLANILMASI
TCK MADDE 245.

[1] Başkasına ait bir banka veya kredi kartını, her ne suretle olursa olsun ele geçiren veya elinde bulunduran kimse, kart sahibinin veya kartın kendisine verilmesi gereken kişinin rızası olmaksızın bunu kullanarak veya kullandırtarak kendisine veya başkasına yarar sağlarsa, üç yıldan altı yıla kadar hapis ve beşbin güne kadar adlî para cezası ile cezalandırılır.
[2] Başkalarına ait banka hesaplarıyla ilişkilendirilerek sahte banka veya kredi kartı üreten, satan, devreden, satın alan veya kabul eden kişi üç yıldan yedi yıla kadar hapis ve onbin güne kadar adlî para cezası ile cezalandırılır.
[3] Sahte oluşturulan veya üzerinde sahtecilik yapılan bir banka veya kredi kartını kullanmak suretiyle kendisine veya başkasına yarar sağlayan kişi, fiil daha ağır cezayı gerektiren başka bir suç oluşturmadığı takdirde, dört yıldan sekiz yıla kadar hapis ve beşbin güne kadar adlî para cezası ile cezalandırılır.

BTK
BİLGİ TEKNOLOJİLERİ VE İLETİŞİM KURUMU

ERİŞİMİN ENGELLENMESİ

IP adresinden erişimin engellenmesi

Alan adından erişimin engellenmesi

URL'den erişimin engellenmesi

Erişimin engellenmesi, farklı filtreleme teknikleri kullanarak İnternet ortamında yer alan herhangi bir yayına kullanıcıların erişmesini önlemeyi ifade etmektedir

Hiçbir erişim engelleme tekniği yüzde yüz başarı sağlamaz. **Kesin çözüm içeriğin çıkarılmasıdır.**

URL adresi (link): Herhangi bir içeriğin (metin, resim, video, müzik, oyun vb.) İnternette bulunduğu tam İnternet adresini ifade eder.

Bu çoğu zaman bir web sitesi içindeki herhangi bir web sayfasının adresidir.

Örneğin: https://www.btk.gov.tr/tr-TR/Yer-Saglayici-Firma-Listesi

BTK
BİLGİ TEKNOLOJİLERİ VE İLETİŞİM KURUMU

Adli mercilerin talepleri (Soruşturma aşamasında hakim, kovuşturma aşamasında mahkeme, GSBH'de savcı)

Vatandaş ihbarları / www.ihbarweb.org.tr

Kamu kurum ve kuruluşlarından gelen ihbarlar / bim.org.tr

Adli
İYM

İdari
İhbarweb

HIS-Operatör

Operatör
(Teknik Değerlendirme)

Hukuk
(Hukuki Değerlendirme)

Erişim Sağlayıcılar
Erişim Engelleme

YURTİÇİ

Resen veya Mahkeme Kararı

* Çocukların Cinsel İstismarı
* Müstehcenlik
* Fuhuş

Mahkeme Kararı

*Atatürk Aleyhine İşlenen Suçlar
* İntihara Yönlendirme
* Uyuşturucu veya Uyarıcı Madde Kullanılmasını Kolaylaştırma
* Sağlık İçin Tehlikeli Madde Temini
* Kumar Oynanması İçin Yer ve İmkan Sağlama

YURTDIŞI

(RESEN veya Mahkeme Kararı)

*Atatürk Aleyhine İşlenen Suçlar
* İntihara Yönlendirme
* Uyuşturucu veya Uyarıcı Madde Kullanılmasını Kolaylaştırma
* Sağlık İçin Tehlikeli Madde Temini
* Kumar Oynanması İçin Yer ve İmkan Sağlama
* Çocukların Cinsel İstismarı
* Müstehcenlik
* Fuhuş

8/A Maddesinin 'Gecikmesinde Sakınca Bulunan Hal' Kapsamında Uygulanması

Madde Kapsamında ihlalin gerçekleştiğini düşünen **Kurumlar** doğrudan BTK'ya başvurarak içerik çıkarma ve/veya erişim engelleme talebinde bulunabilir.

Computer Networks

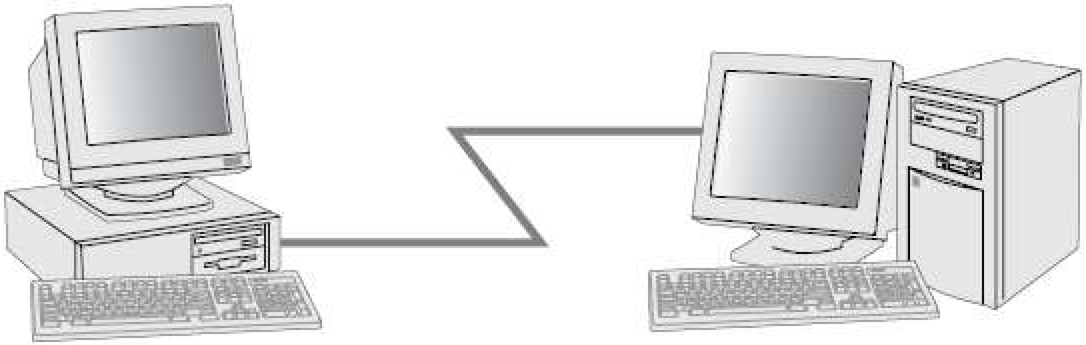
BBM 105

Slide Credit: Assoc. Prof. Sevil Şen

What is a Network?

- A network consists of 2 or more computers connected together, and they can communicate and share resources (e.g. information)

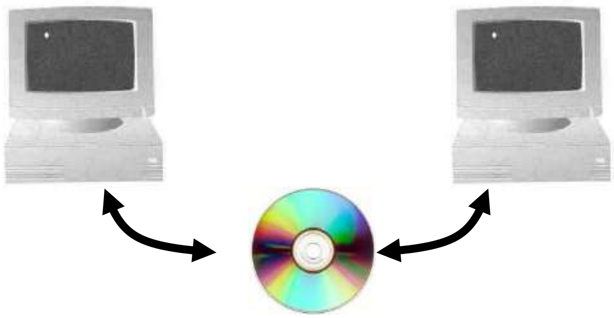
- A small network could be as simple as two computers linked together by a single cable

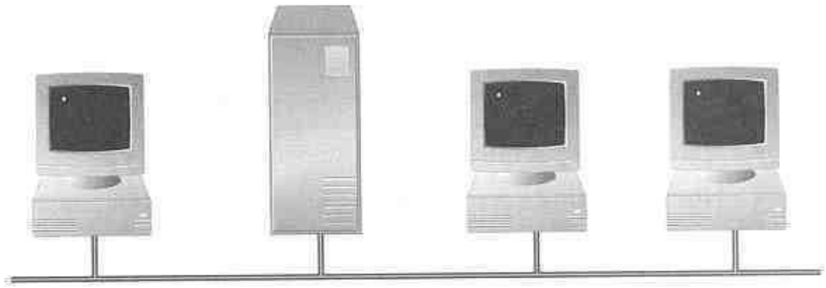


Why Networking?

- Sharing Information

Which one do you prefer?





Why Networking?

- Resource Sharing

  equipment (*e.g.* printer), data (*e.g.* business information)

- Communication Medium

  (electronic mail, video conferencing)

- High reliability

  distributing redundant copies of data over the network

  accessing data via different paths

Different Kinds of Networks

We can classify networks in different ways:

- Based on transmission media

  Wired(UTP, coaxial, fiber-optic)

  Wireless

- Based on network size: LAN and WAN

- Based on management method:

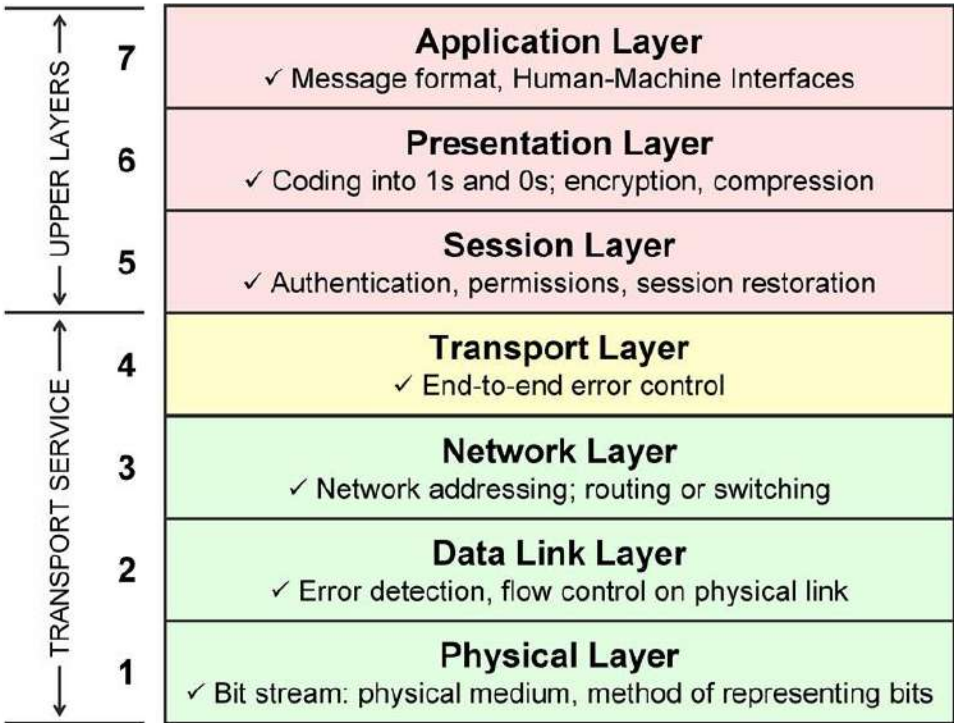
  Peer-to-peer and Client/Server

- Based on topology (connectivity): Bus, Star, Ring …

OSI Layers

Application Layer
High-level APIs, including resource sharing, remote file access

Presentation Layer
Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption (XML,HTML)



Session Layer
Managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes (HTTP, SSH)

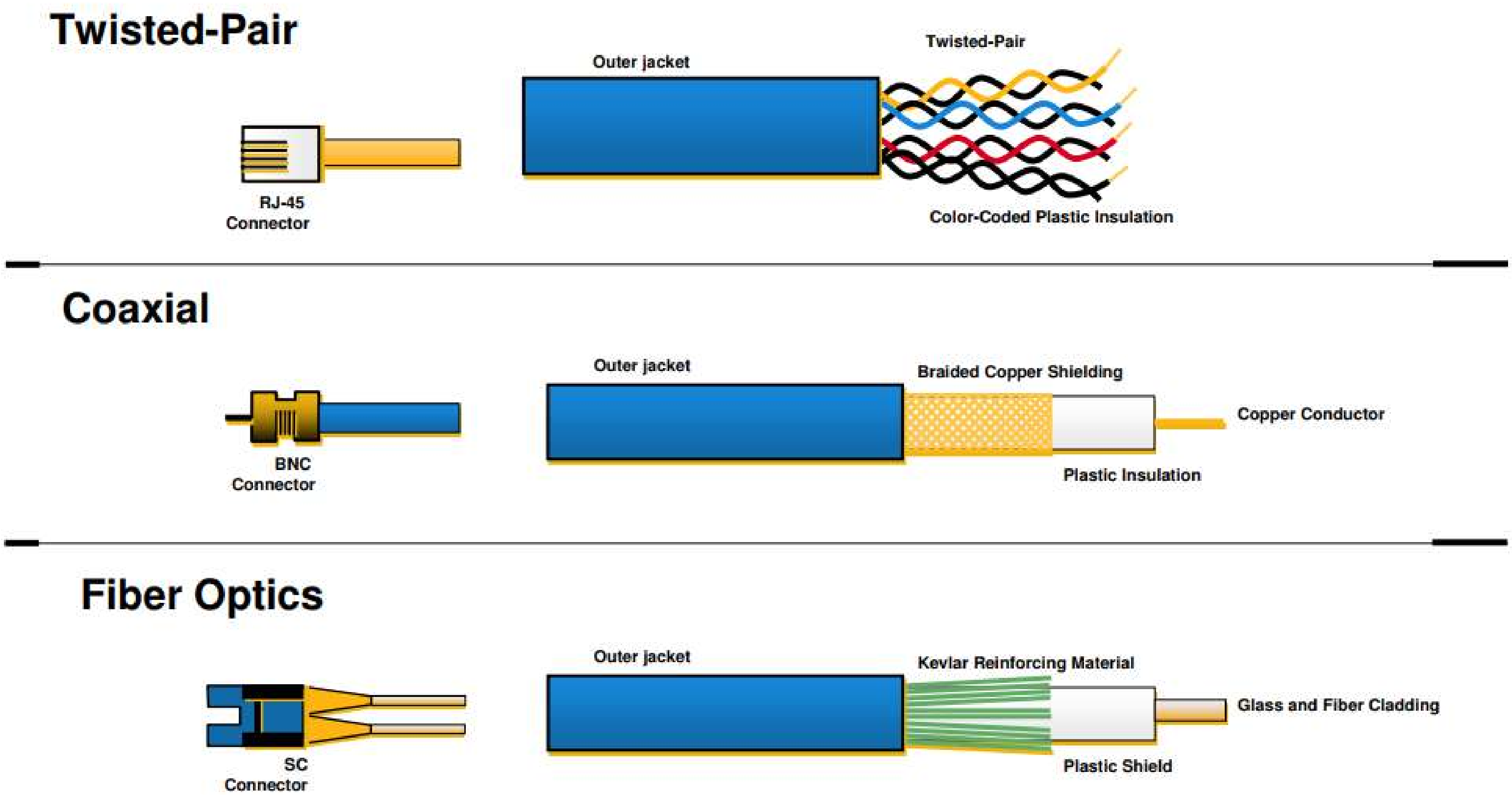
Transport Layer
Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing (TCP,UDP)

Physical Layer
Transmission and reception of raw bit streams over a physical medium(DSL,Ethernet)

Data Link Layer
Reliable transmission of data frames between two nodes connected by a physical layer (MAC)

Network Layer
Structuring and managing a multi-node network, including addressing, routing and traffic control (IPv4, IPv6)

Physical Media Types



Local Area Networks (LAN)

- Small network, short distance
  - A room, a floor, a building
  - Limited by number of computers and distance covered
  - Usually one kind of technology throughout the LAN
  - Serve a department within an organization

- Examples
  - Network inside your house, your room

Wide Area Networks (WAN)

- A network that uses long-range telecommunication links to connect 2 or more LANs/computers located in different places far apart.

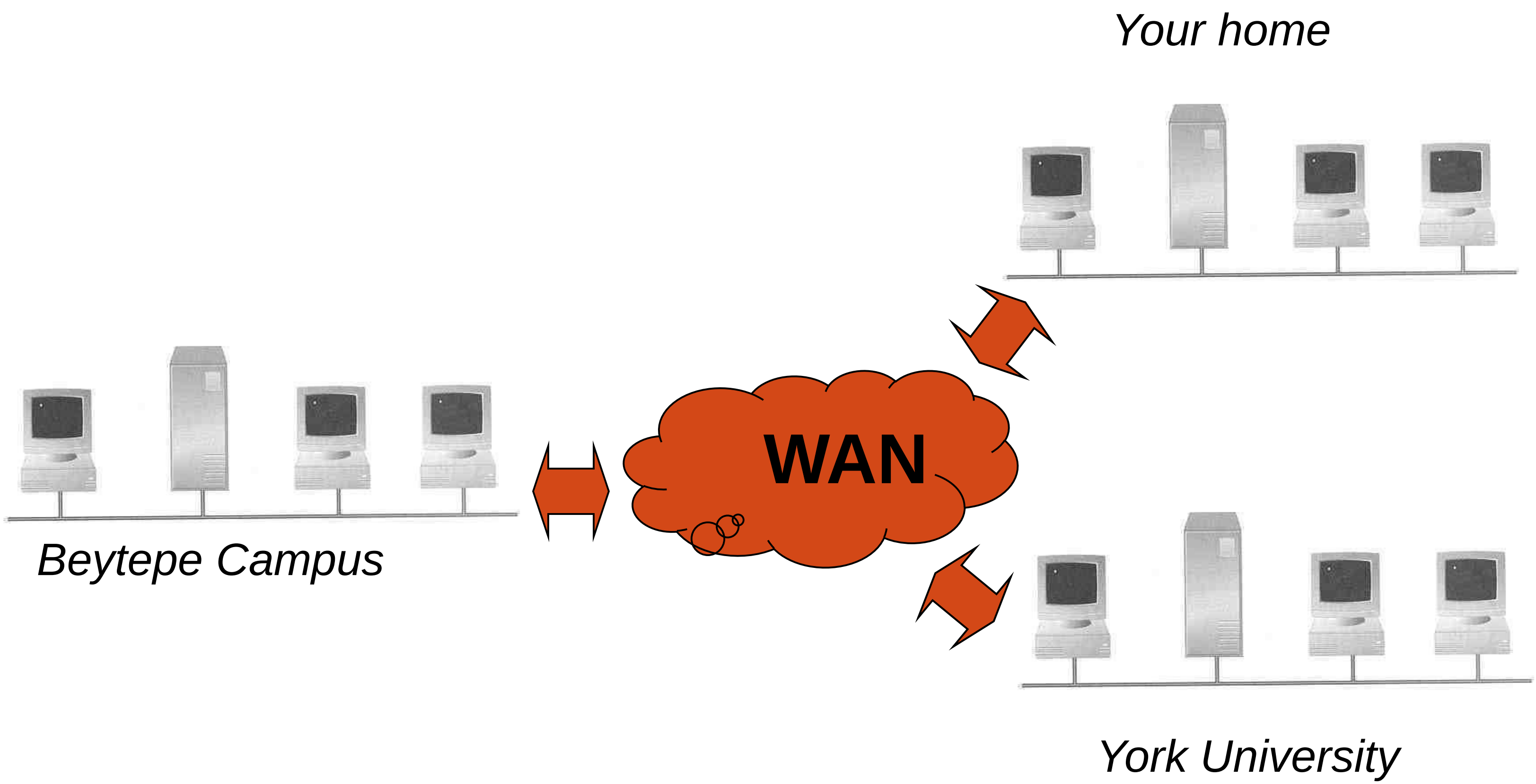
  Town, states, countries

- Examples

  Network of our campus

  Internet

An Example WAN

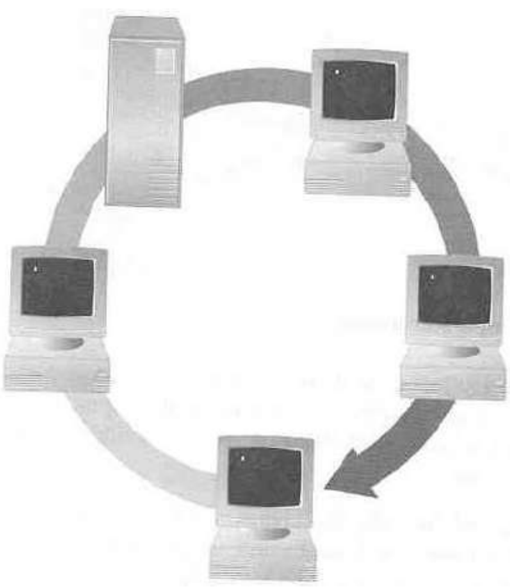


LAN Topologies

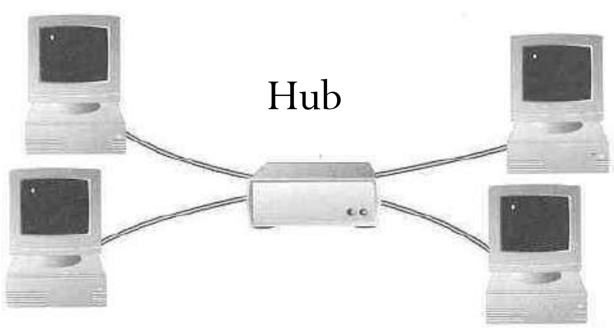
How so many computers are connected together?

Bus Topology

Ring Topology



Star Topology



Bus Topology

Bus Topology

- Simple and low-cost
- A single cable called a trunk (backbone, segment)
- Only one computer can send messages at a time
- Passive topology - computer only listen for, not regenerate data

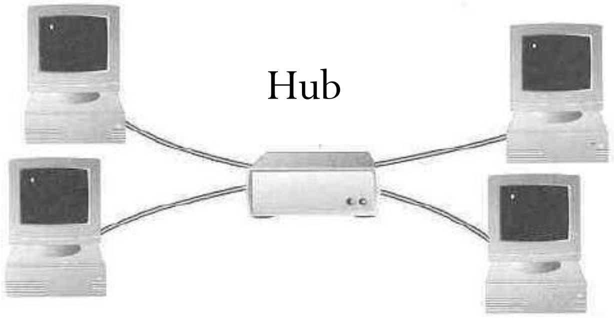
Star Topology

Star Topology

- Each computer has a cable connected to a single point
- More cabling, hence higher cost
- All signals transmission through the hub; if down, entire network down
- Depending on the intelligence of hub, two or more computers may send message at the same time

A typical hub consist of an electronic device that accepts data from a sending computer and broadcasts to all computers.

Switch is more developed kind of device than hub. It delivers the data to the appropriate destination



Ring Topology

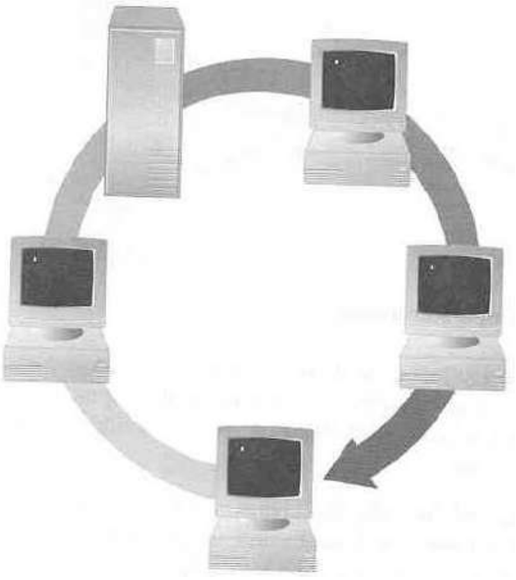
Every computer serves as a repeater to boost signals

Typical way to send data: *Token passing*

only the computer who gets the token can send data

Disadvantages

- Difficult to add computers
- More expensive
- If one computer fails, whole network fails



Packet

Most computer networks divides data into small blocks called packets

Computer networks are called *packet networks* or *packet switching network*

1 Byte = 8 bit

1 KB (kilobyte) = 1024 byte

1 MB (megabyte) = 1024 KB

1GB (gigabyte) = 1024 KB

Example :

a 5MB file

the communication system can transfer 56.000 bits per second.

Network Address

A unique identifier for a computer on a network

- Computer can determine the addresses of other computers on the network
- Computer use these addresses to send messages to each other.

Internet Protocol (IP) addresses uniquely identify all computers on the public Internet.

IPv4 4bytes (32 bits)

IPv6 6bytes

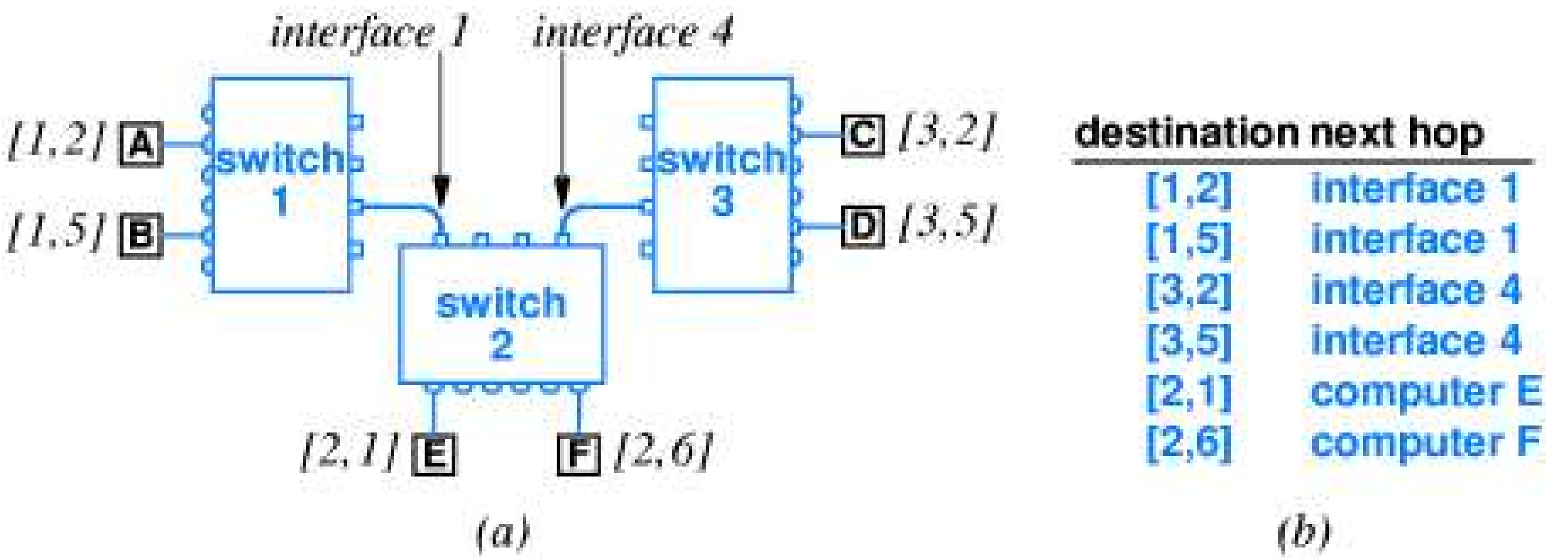
Media Access Control (MAC) address (physical address)

six bytes (48 bits)

manufacturers of network adapters burn into their products

unique

Switch

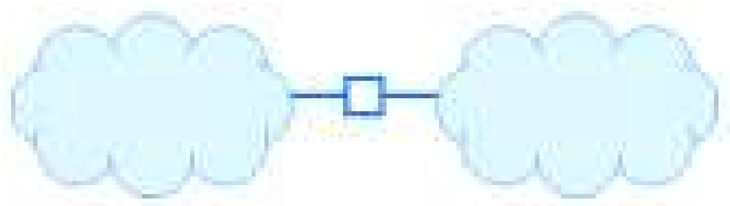


<u>Packet switch</u>

not keep complete information about how to reach all possible destinations.

has information about the next hop.

Network Connection with Routers

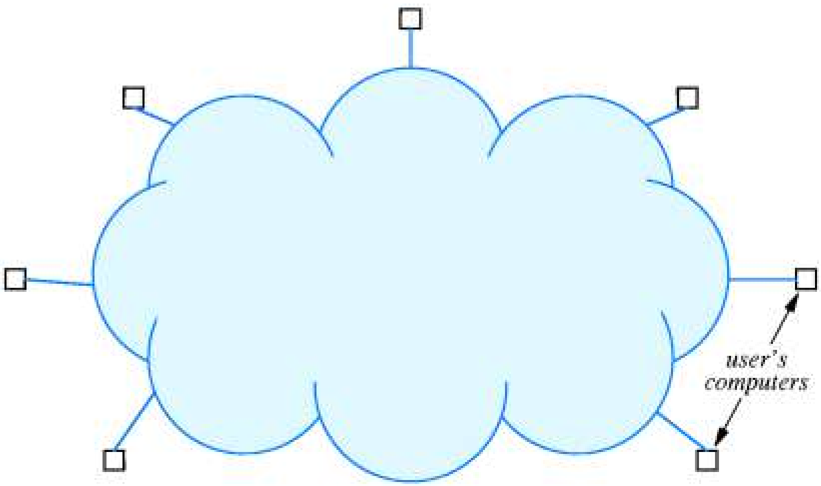


Two physical networks connected by a router

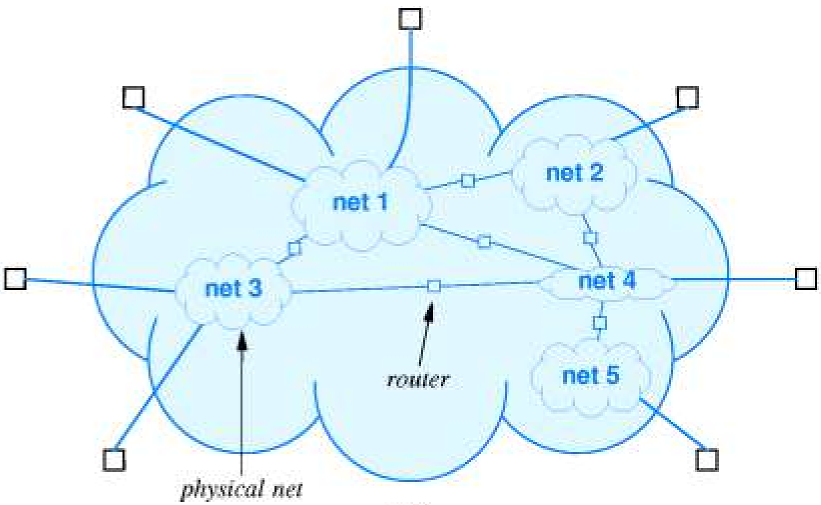
Router is a special-purpose system dedicated to the task of interconnecting network.

Interconnect networks with different technologies, media, etc.

Internet


user's computers

(a)


net 1
net 2
net 3
net 4
net 5
router
physical net

(b)

Client-Server

<u>Clients</u>

Computers that request network resources or services

<u>Servers</u>

- Computers that manage and provide network resources and services to clients
- Usually have more processing power, memory and hard disk space than clients
- Run a system that can manage not only data, but also users, groups, security, and applications on the network
- Servers often have a more stringent requirement on its performance and reliability

Client-Server

- Functions such as email exchange, web access and database access, are built on the client–server model.

- *Example:* Users accessing banking services from their computer use a web browser client to send a request to a web server at a bank.

- The client–server model has become one of the central ideas of network computing.

  Many business applications

  The Internet's main application protocols,

  such as HTTP, SMTP, Telnet, and DNS.

Client-Server

Advantages

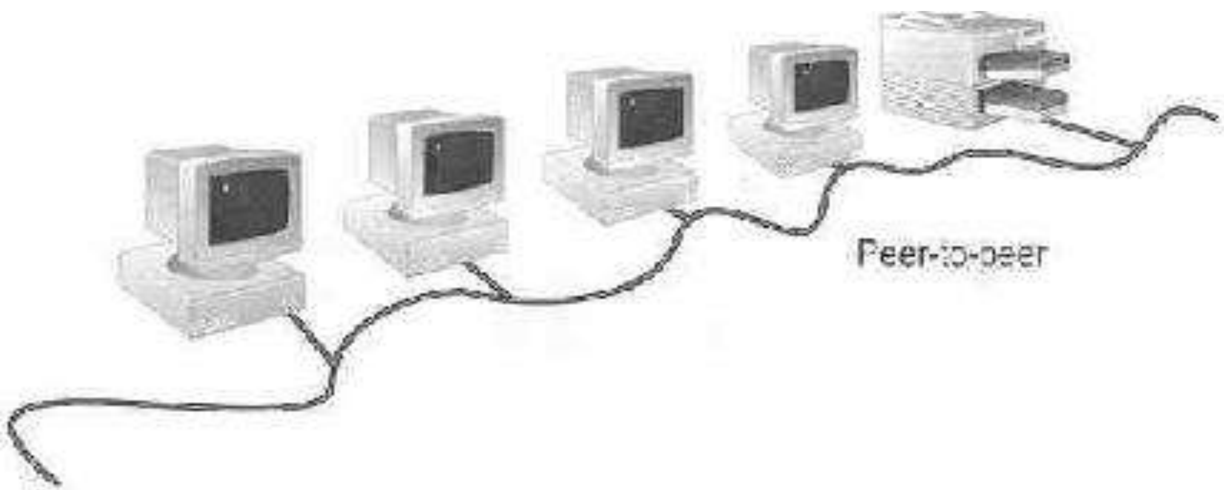
- Facilitate resource sharing – centrally administrate and control
- Facilitate system backup and improve fault tolerance
- Enhance security – only administrator can have access to Server
- Support more users – difficult to achieve with peer-to-peer networks

Disadvantages:

- High cost for Servers
- Need expert to configure the network
- Introduce a single point of failure to the system

Peer-to-Peer Networks (P2P)

- No hierarchy among computers ⇒ all are equal
- No administrator responsible for the network
- The peer-to-peer application structure was popularized by file sharing systems like Napster



Peer-to-Peer Networks (P2P)

Advantages

- Low cost
- Simple to configure
- User has full accessibility of the computer

Disadvantages

- May have duplication in resources
- Difficult to uphold security policy
- Difficult to handle uneven loading

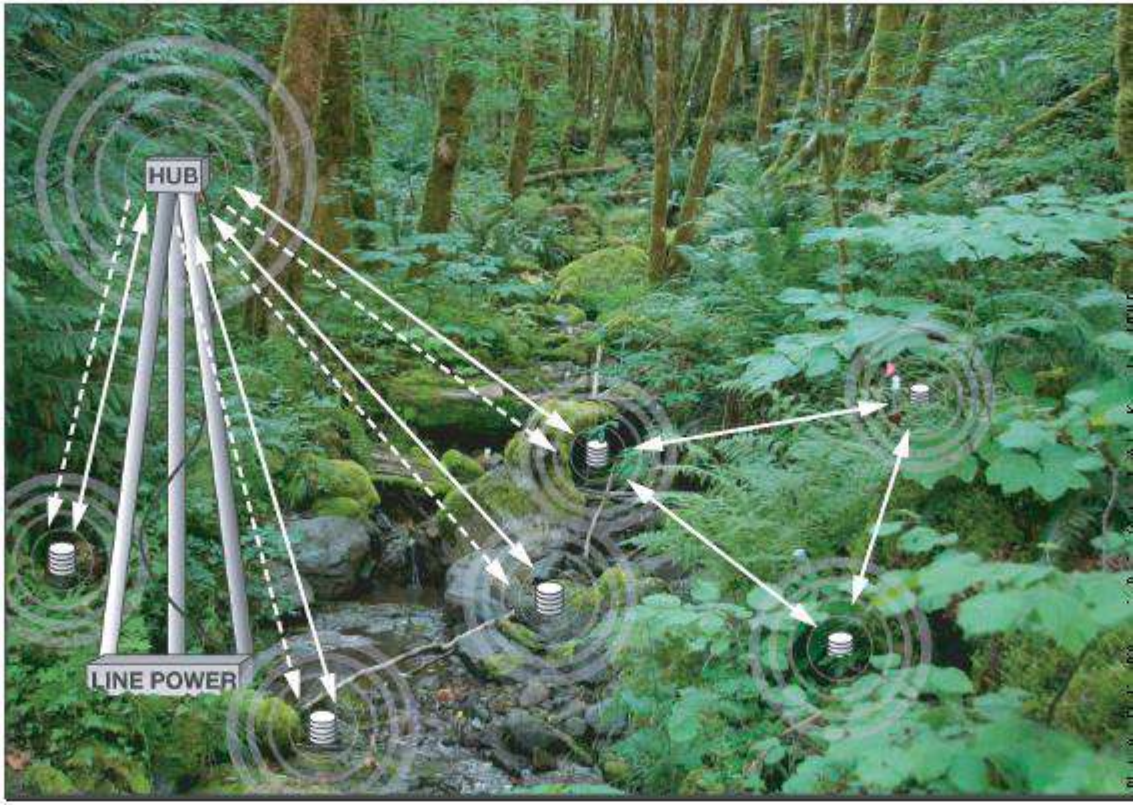
Where peer-to-peer network is appropriate:

- 10 or less users
- No specialized services required
- Security is not an issue
- Only limited growth in the foreseeable future

Wireless Networks
(Kablosuz Ağlar)

Wireless Sensor Networks (WSNs)

Consists of spatially distributed autonomous sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, pressure, motion or pollutants.

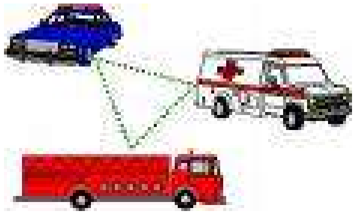


Mobile Ad Hoc Networks (MANETs)

Self configuring network of mobile nodes connected by wireless links

Provide communication in the absence of a fixed infrastructure

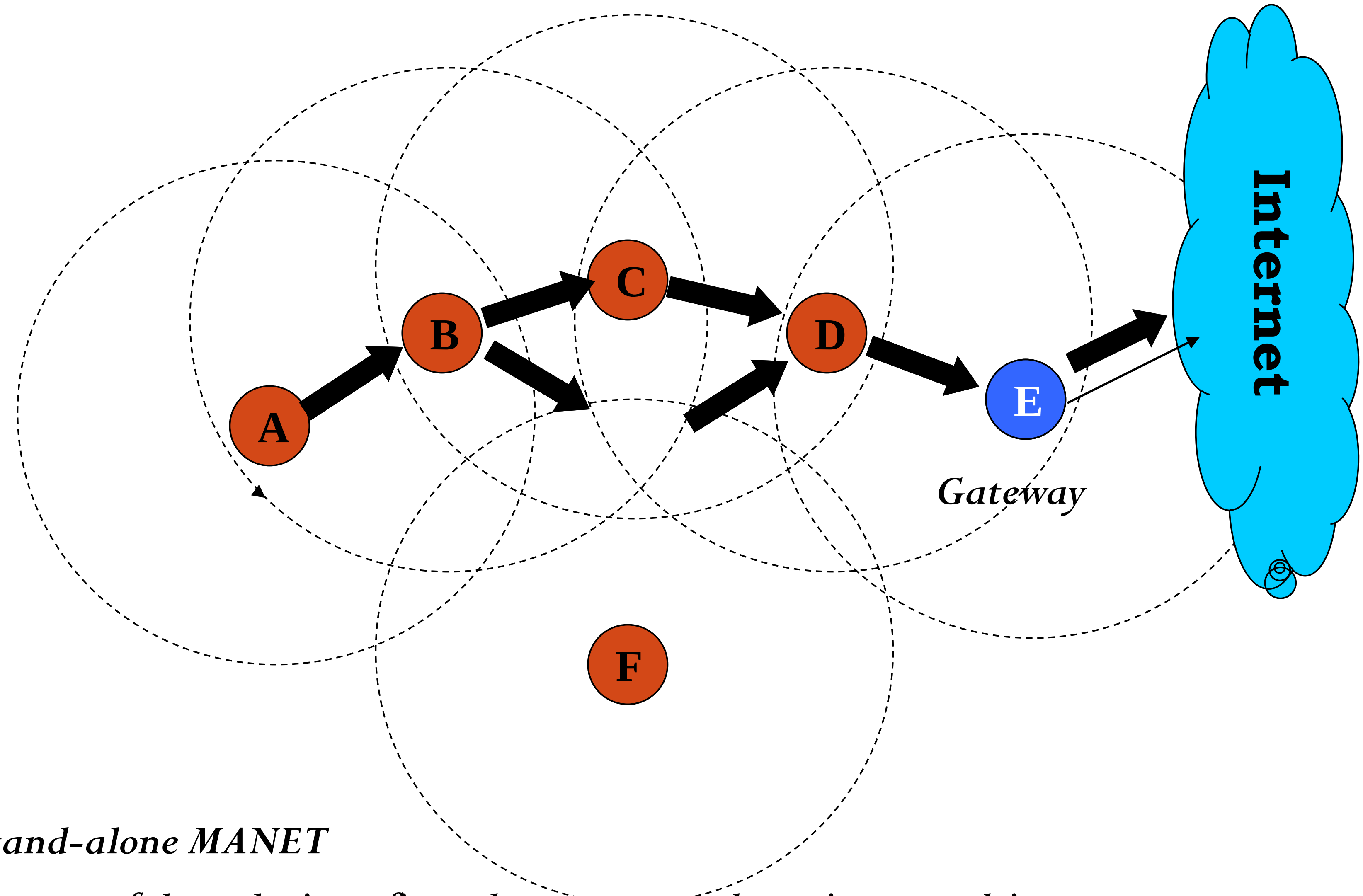
Dynamic topology due to mobility



Attractive for many applications

- disaster recovery operations
- military applications

MANETs: Operation



Stand-alone MANET

After one of the nodes is configured as a gateway, the entire network is connected to an external network like Internet

Vehicular Ad Hoc Networks (VANETs)

- moving cars as nodes in a network
- every participating car is a wireless router or node
- allowing cars approximately 100 to 300 metres of each

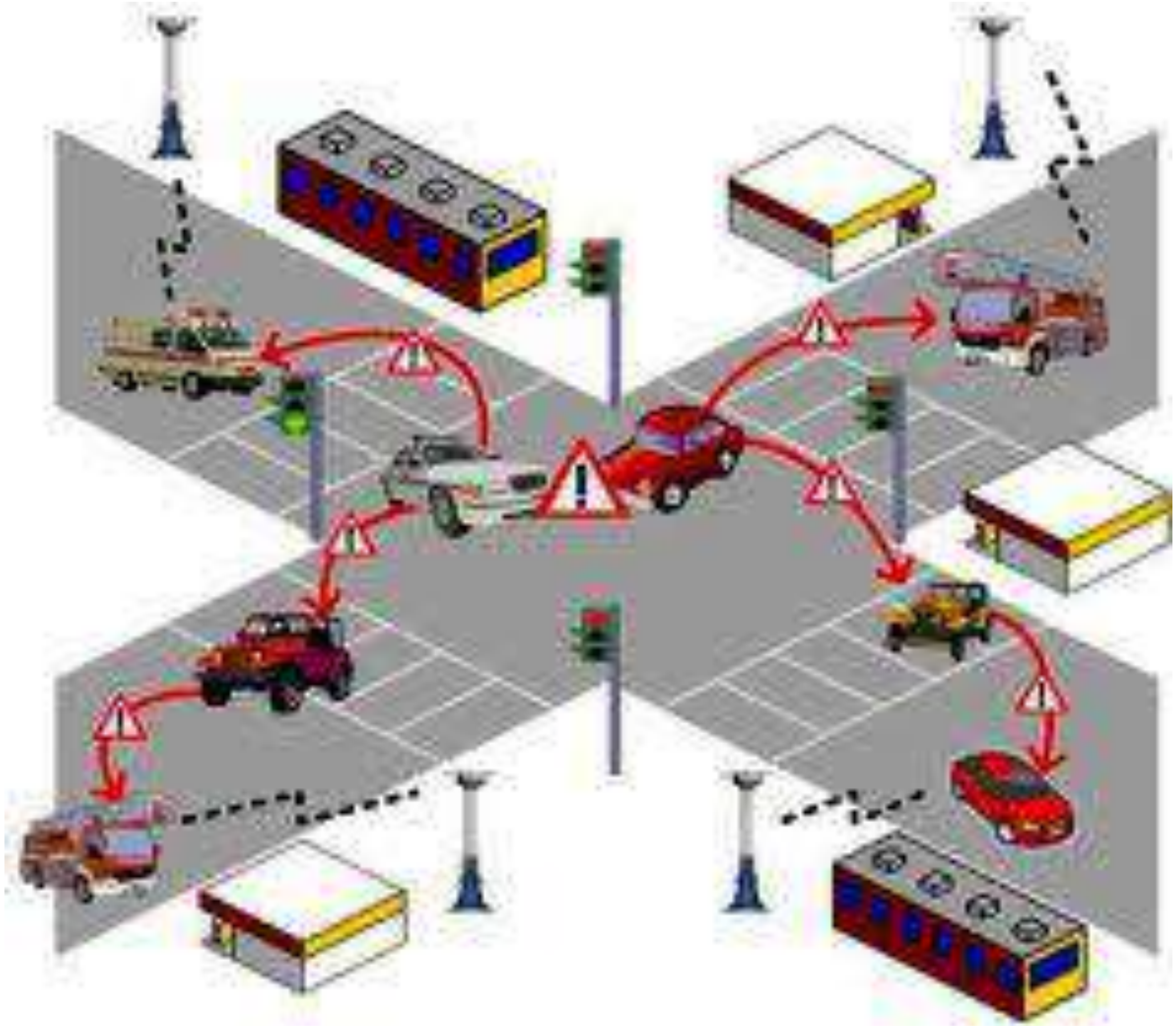
other to connect and, create a network with a wide range

- a mobile Internet is created.

Future applications:

police and fire vehicles

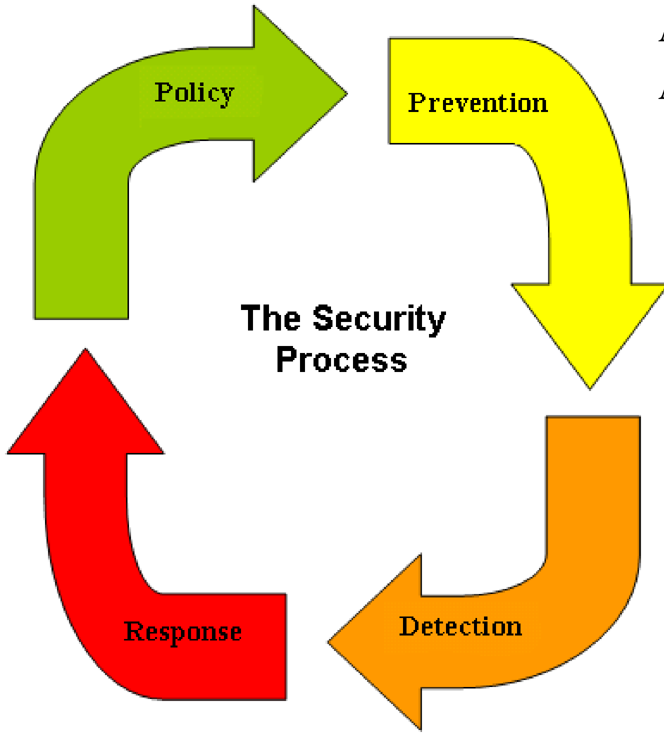
to communicate with each other for safety purposes



Network Security

Plans, rules,
constraints…

Access control
Authentication



Monitoring
Intrusion Detection

Active/Passive Response
Alarm to proper authority, ..

BBM105 - Databases

Slides: Invitation to Computer Science 5th Edition (Chapter 14)

Databases

- Bit
  - Most basic unit of data
  - Combined into groups of eight called bytes
- Fields
  - Group of bytes
- Record
  - Collection of related fields

Databases (continued)

- Data file
  - Stores related records
- Database
  - Made up of related files

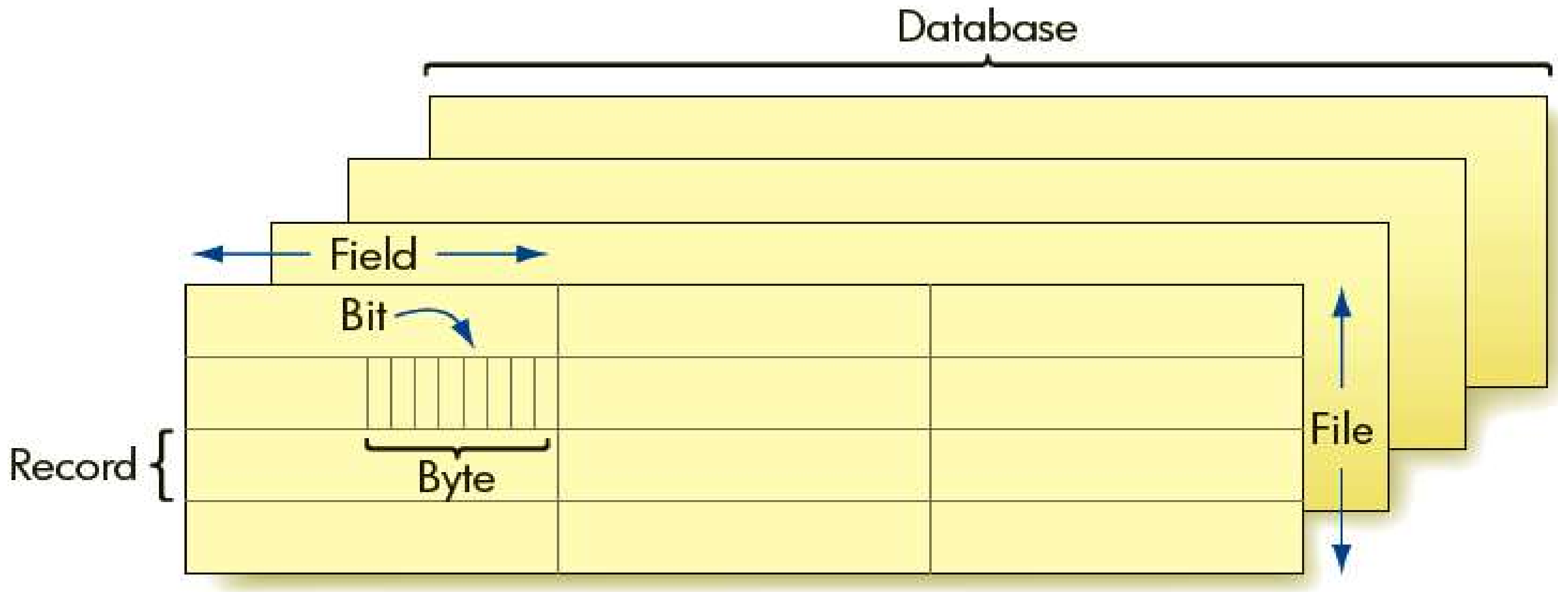


Figure 14.3 Data Organization Hierarchy

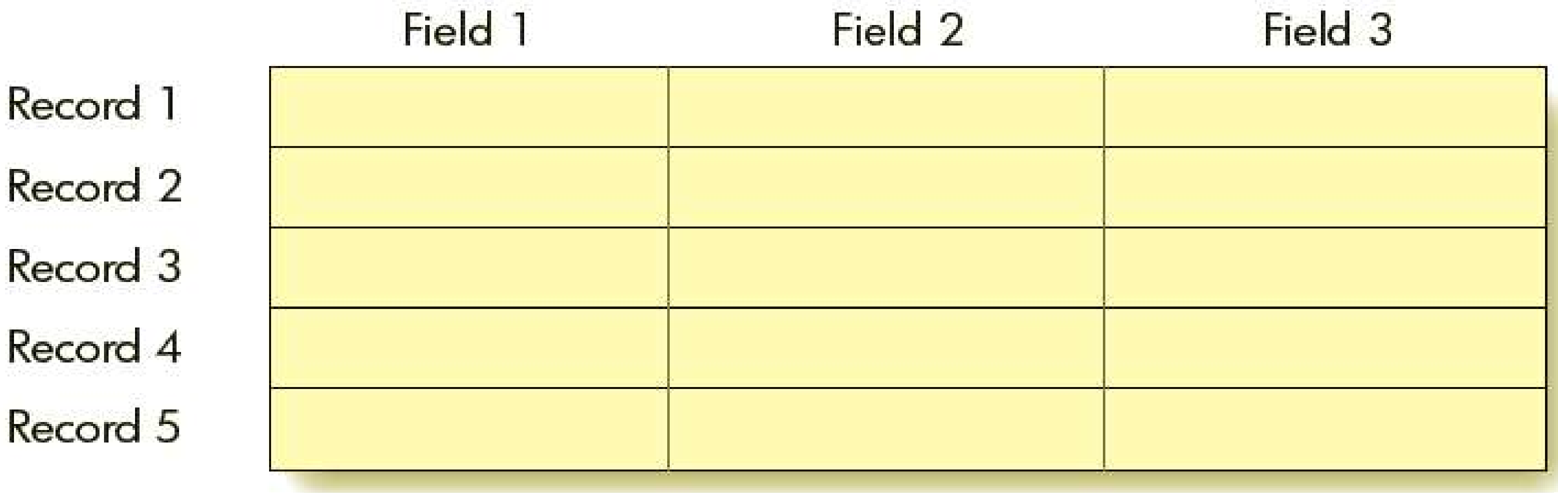


Figure 14.4 Records and Fields in a Single File

ID	LastName	FirstName	Birthdate	PayRate	HoursWorked
149	Takasano	Frederick	5/23/1966	$12.35	250

Figure 14.5 One Record in the Rugs-For-You Employees File

Database Management Systems

- Manage the files in a database
- Entity
  - Fundamental distinguishable component
- Attribute
  - Category of information
- Primary key
  - Attribute or combination of attributes that uniquely identifies a **tuple**

Employees

ID	LastName	FirstName	Birthdate	PayRate	HoursWorked
116	Kay	Janet	3/29/1956	$16.60	94
123	Perreira	Francine	8/15/1987	$ 8.50	185
149	Takasano	Frederick	5/23/1966	$12.35	250
171	Kay	John	11/17/1954	$17.80	245
165	Honou	Morris	6/9/1988	$ 6.70	53

Figure 14.6 Employees Table for Rugs-For-You

Database Management Systems (continued)

- Query languages
  - Enable user or another application program to **query** the database, in order to retrieve information
- Composite primary key
  - Needed to identify a tuple uniquely
- Foreign key
  - Key from another table that refers to a specific key, usually the primary key

Database Systems

- The big commercial database vendors:
  - Oracle
  - IBM (with DB2)
  - Microsoft (SQL Server)
  - Sybase
- Some free database systems:
  - Postgres
  - MySQL
  - Predator

InsurancePolicies

EmployeeID	PlanType	DateIssued
171	B2	10/18/1974
171	C1	6/21/1982
149	B2	8/16/1990
149	A1	5/23/1995
149	C2	12/18/1999

Figure 14.7 Insurance Policies Table for Rugs-For-You

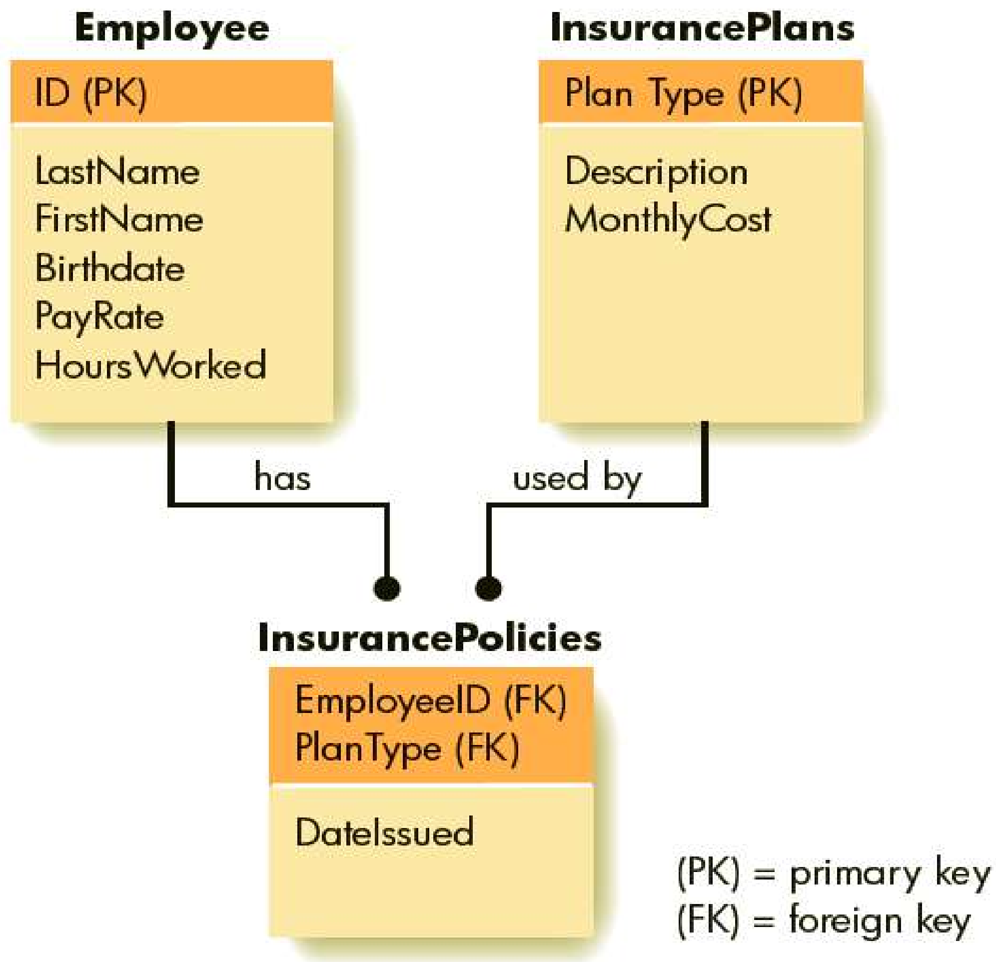


Figure 14.8 Three Entities in the Rugs-For-You Database

Functionality of a DBMS

The programmer sees SQL, which has two components:

- Data Definition Language - DDL
- Data Manipulation Language - DML
  - query language (Select, Insert, Delete, Update)

Behind the scenes the DBMS has:

- Query engine
- Query optimizer
- Storage management
- Transaction Management (concurrency, recovery)

How the Programmer Sees the DBMS

- Tables:

SSN	Name	Category
123-45-6789	Charles	undergrad
234-56-7890	Dan	grad
	…	…

SSN	CID
123-45-6789	CSE444
123-45-6789	CSE444
234-56-7890	CSE142
	…

CID	Name	Quarter
CSE444	Databases	fall
CSE541	Operating systems	winter

- Still implemented as files, but behind the scenes can be quite complex

Queries

- Find all courses that “Mary” takaes
  - ```
    SELECT  C.name
    FROM    Students S, Takes T, Courses C
    WHERE   S.name="Mary" and
            S.ssn = T.ssn and T.cid = C.cid
    ```
- What happens behind the scene ?
  - Query processor figures out how to answer the query efficiently.
    ```

Transactions

- A *transaction* = sequence of statements that either all succeed, or all fail
- Transactions have the ACID properties:

  A = atomicity (a transaction should be done or undone completely)

  C = consistency (a transaction should transform a system from one consistent state to another consistent state)

  I = isolation (each transaction should happen independently of other transactions)

  D = durability (completed transactions should remain permanent)

Other Considerations

- Performance issues
  - Affect the user’s satisfaction with a database management system
- To significantly reduce access time:
  - Create additional records to be stored along with the file
- Distributed databases
  - Allow the physical data to reside at separate and independent locations that are electronically networked together

Summary

- E-business
  - Every part of a financial transaction is handled electronically
- Opening an online store
  - Requires a significant amount of planning
- Database
  - Allows data items to be stored, extracted, sorted, and manipulated
- Relational database model
  - Conceptual model of a file as a two-dimensional table

BBM105: Binary Numbers, Boolean Logic, and Gates

Slides From: Invitation to Computer Science, C++ Version

Objectives

In this course, you will learn about:

- The binary numbering system
- Boolean logic and gates
- Building computer circuits
- Control circuits

The Binary Numbering System

- A computer's internal storage techniques are different from the way people represent information in daily lives

- Information inside a digital computer is stored as a collection of **binary data**

Binary Representation of Numeric and Textual Information

- Binary numbering system
  - Base-2
  - Built from ones and zeros
  - Each position is a power of 2

    $1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$
- Decimal numbering system
  - Base-10
  - Each position is a power of 10

    $3052 = 3 \times 10^3 + 0 \times 10^2 + 5 \times 10^1 + 2 \times 10^0$

Binary Representation of Numeric and Textual Information (continued)

- Representing integers
  - Decimal integers are converted to binary integers
  - Given k bits, the largest unsigned integer is $2^k - 1$
    - Given 4 bits, the largest is $2^4-1 = 15$

Binary Representation of Numeric and Textual Information (continued)

- Signed integers must also represent the sign (positive or negative) - ***Sign/Magnitude notation***
  - **0**000 = +0
  - 0001 = +1
  - 0010 = +2
  - ...
  - **1**000 = -0
  - 1001 = -1
  - 1111 = -7

Binary Representation of Numeric and Textual Information (continued)

- Signed integers must also represent the sign (positive or negative) – ***Two's Complement!!***
  - Convert Decimal to Two's Complement (to 4 bit)
    - e.g. We have 6 and -6 decimal numbers.
    - For 6 → 0110. It's ok for positive numbers!
    - For -6 → we have posivitive 0110.
      - In 2nd step, invert all the bits 0110 → 1001
      - In 3rd step, add 1 to 1001 → 1010. Now we have -6!

Binary Representation of Numeric and Textual Information (continued)

- Signed integers must also represent the sign (positive or negative) – ***Two's Complement!!***
  - Convert Two's Complement to Decimal(to 4 bit)
    - e.g. We have 0110 and 1010 decimal numbers.
    - For 0110 → First bit is 0. So this is a positive number!
      - 0110 = 0x2^3 + 1x 2^2 + 1x2^1+0x2^0 = 6
    - For 1010 → First bit is 1. So this is a negative number!
      - Invert all bits again 1010 → 0101
      - Add 1 to the results 0101 + 1 → 0110
      - 0110 = 0x2^3 + 1x 2^2 + 1x2^1+0x2^0 = 6
      - The number was negative, because of first bit! = -6

Binary Representation of Numeric and Textual Information (continued)

- Characters are mapped onto binary numbers
  - ASCII code set
    - 8 bits per character; 256 character codes
  - UNICODE code set
    - 16 bits per character; 65,536 character codes
- Text strings are sequences of characters in some encoding

Binary Representation of Textual Information (cont'd)

ASCII
8 bits long

Decimal	Binary	Val.
48	00110000	0
49	00110001	1
50	00110010	2
51	00110011	3
52	00110100	4
53	00110101	5
54	00110110	6
55	00110111	7
56	00111000	8
57	00111001	9
58	00111010	:
59	00111011	;
60	00111100	<
61	00111101	=
62	00111110	>
63	00111111	?
64	01000000	@
65	01000001	A
66	01000010	B

Dec.	Unicode	Charac.
0x30	0x0030	[0]
0x31	0x0031	[1]
0x32	0x0032	[2]
0x33	0x0033	[3]
0x34	0x0034	[4]
0x35	0x0035	[5]
0x36	0x0036	[6]
0x37	0x0037	[7]
0x38	0x0038	[8]
0x39	0x0039	[9]
0x3A	0x003A	[:]
0x3B	0x003B	[;]
0x3C	0x003C	[<]
0x3D	0x003D	[=]
0x3E	0x003E	[>]
0x3F	0x003F	[?]
0x40	0x0040	[@]
0x41	0x0041	[A]
0x42	0x0042	[B]

Unicode
16 bits long

Partial listings only!

Binary Representation of Images

- Representing image data
  - Images are sampled by reading color and intensity values at even intervals across the image
  - Each sampled point is a pixel
  - Image quality depends on number of bits at each pixel

Binary Representation of Images (cont'd)

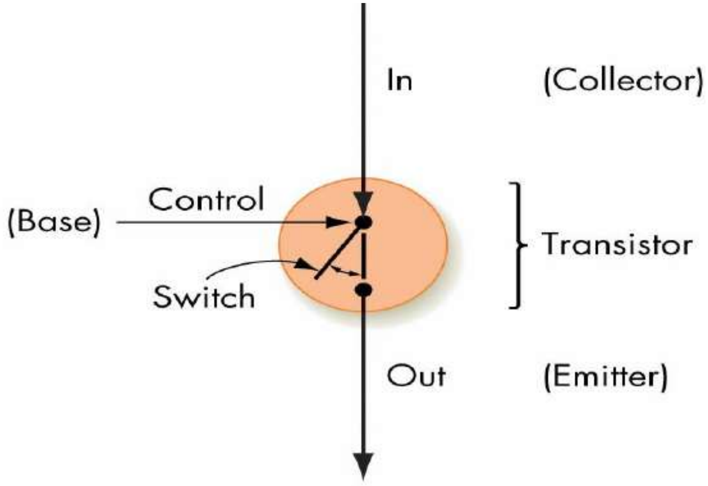
- Representing image data
  - Images are sampled by reading color and intensity values at even intervals across the image
  - Each sampled point is a pixel
  - Image quality depends on number of bits at each pixel
  - More image information: http://cat.xula.edu/tutorials/imaging/grayscale.php

The Reliability of Binary Representation

- Electronic devices are most reliable in a bistable environment
- Bistable environment
  - Distinguishing only two electronic states
    - Current flowing or not
    - Direction of flow
- Computers are bistable: hence binary representations

Binary Storage Devices (continued)

- Transistors
  - Solid-state switches: either permits or blocks current flow
  - A control input causes state change
  - Constructed from semiconductors



Boolean Logic and Gates: Boolean Logic

- Boolean logic describes operations on true/false values
- True/false maps easily onto bistable environment
- Boolean logic operations on electronic signals may be built out of transistors and other electronic devices

Boolean Logic (continued)

- Boolean operations
  - a AND b
    - True only when a is true and b is true
  - a OR b
    - True when either a is true or b is true, or both are true
  - NOT a
    - True when a is false, and vice versa

Boolean Logic (continued)

- Boolean expressions
  - Constructed by combining together Boolean operations
    - Example: (a AND b) OR ((NOT b) AND (NOT a))
- Truth tables capture the output/value of a Boolean expression
  - A column for each input plus the output
  - A row for each combination of input values

Boolean Logic (continued)

- Example:

  (a AND b) OR ((NOT b) and (NOT a))

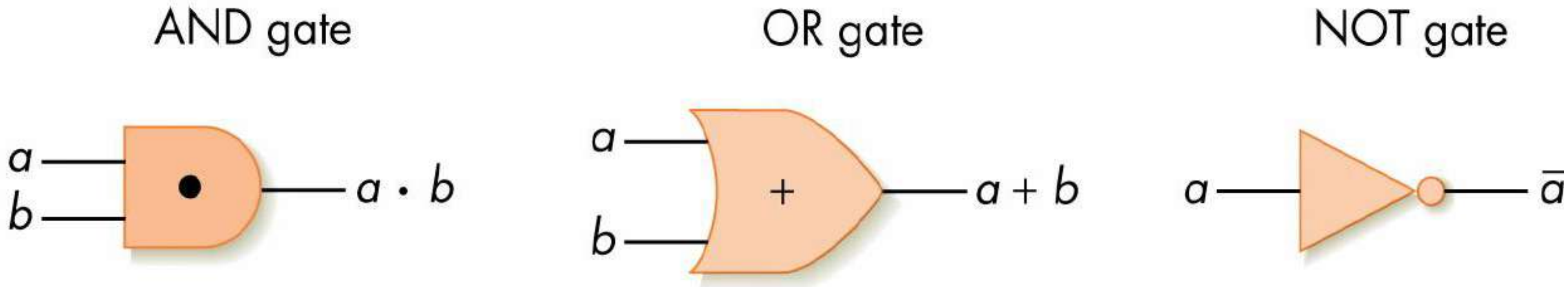
a	*b*	*Value*
0	0	1
0	1	0
1	0	0
1	1	1

Gates

- Gates
  - Hardware devices built from transistors to mimic Boolean logic
- AND gate
  - Two input lines, one output line
  - Outputs a 1 when both inputs are 1

Gates (continued)

- OR gate
  - Two input lines, one output line
  - Outputs a 1 when either input is 1
- NOT gate
  - One input line, one output line
  - Outputs a 1 when input is 0 and vice versa



a	b	a · b
0	0	0
0	1	0
1	0	0
1	1	1

a	b	a + b
0	0	0
0	1	1
1	0	1
1	1	1

a	ā
0	1
1	0

Figure 4.15

The Three Basic Gates and Their Symbols

Gates (continued)

- Abstraction in hardware design
  - Map hardware devices to Boolean logic
  - Design more complex devices in terms of logic, not electronics
  - Conversion from logic to hardware design may be automated

Building Computer Circuits: Introduction

- A circuit is a collection of logic gates:
  - Transforms a set of binary inputs into a set of binary outputs
  - Values of the outputs depend only on the current values of the inputs
- Combinational circuits have no cycles in them (no outputs feed back into their own inputs)

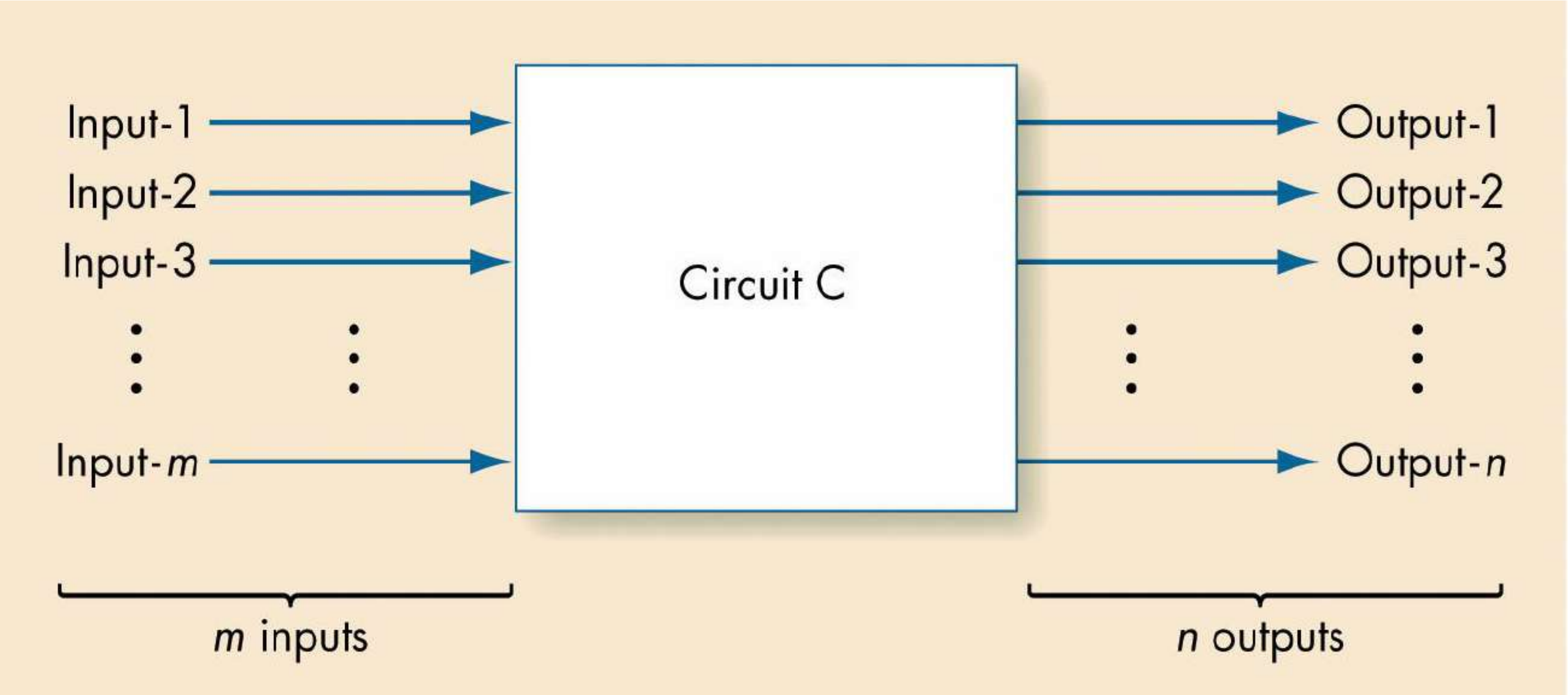


Figure 4.19

Diagram of a Typical Computer Circuit

A Circuit Construction Algorithm

- Sum-of-products algorithm is one way to design circuits:
  - Truth table to Boolean expression to gate layout

Sum-of-Products Algorithm for Constructing Circuits

1. Construct the truth table describing the behavior of the desired circuit
2. While there is still an output column in the truth table, do steps 3 through 6
3. Select an output column
4. Subexpression construction using AND and NOT gates
5. Subexpression combination using OR gates
6. Circuit diagram production
7. Done

Figure 4.21

The Sum-of-Products Circuit Construction Algorithm

A Circuit Construction Algorithm (continued)

- Sum-of-products algorithm
  - Truth table captures every input/output possible for circuit
  - Repeat process for each output line
    - Build a Boolean expression using AND and NOT for each 1 of the output line
    - Combine together all the expressions with ORs
    - Build circuit from whole Boolean expression

A Compare-for-equality Circuit

- Compare-for-equality circuit
  - CE compares two unsigned binary integers for equality
  - Built by combining together 1-bit comparison circuits (1-CE)
  - Integers are equal if corresponding bits are equal (AND together 1-CD circuits for each pair of bits)

A Compare-for-equality Circuit (continued)

- 1-CE circuit truth table

a	*b*	*Output*
0	0	**1**
0	1	0
1	0	0
1	1	**1**

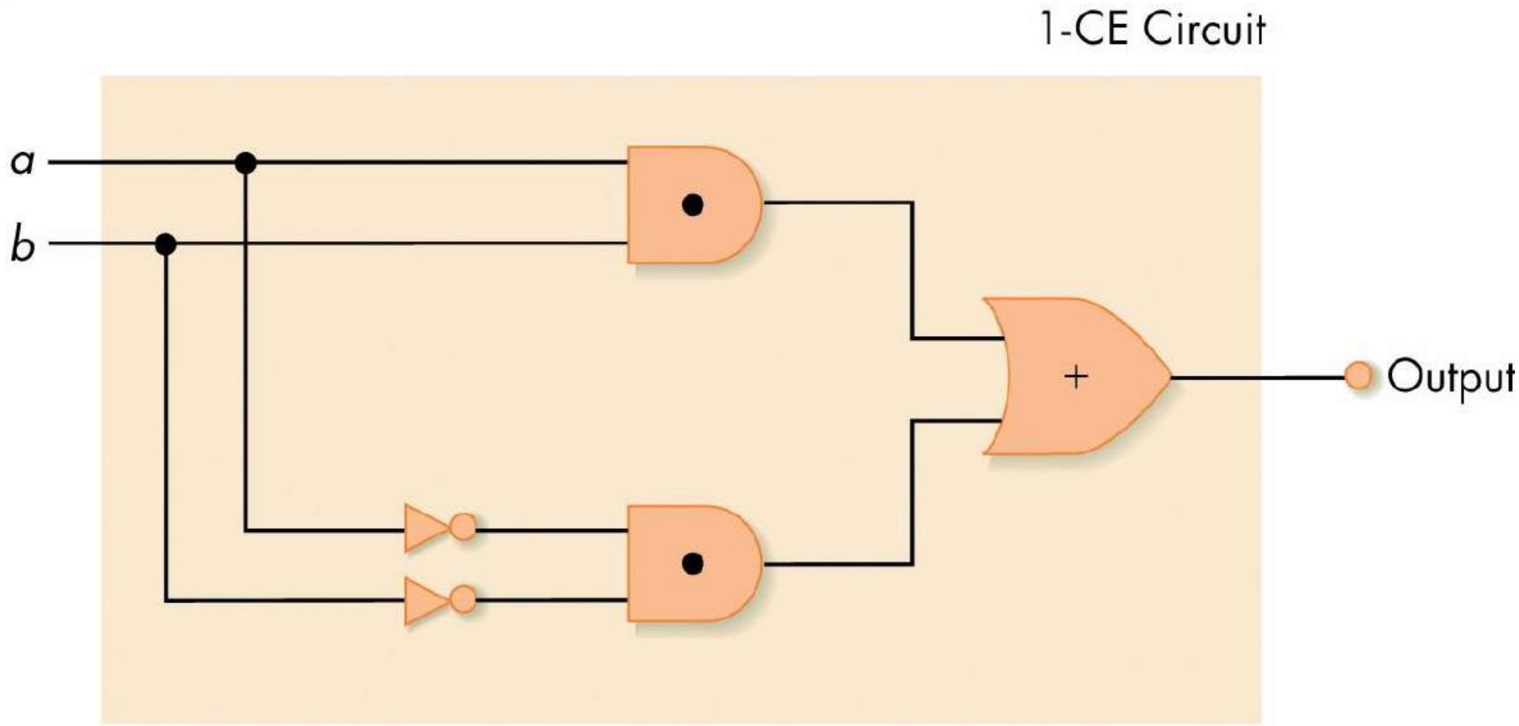


Figure 4.22

One-Bit Compare for Equality Circuit

A Compare-for-equality Circuit (continued)

- 1-CE Boolean expression
  - <u>First case</u>: (NOT a) AND (NOT b)
  - <u>Second case</u>: a AND b
  - <u>Combined</u>:

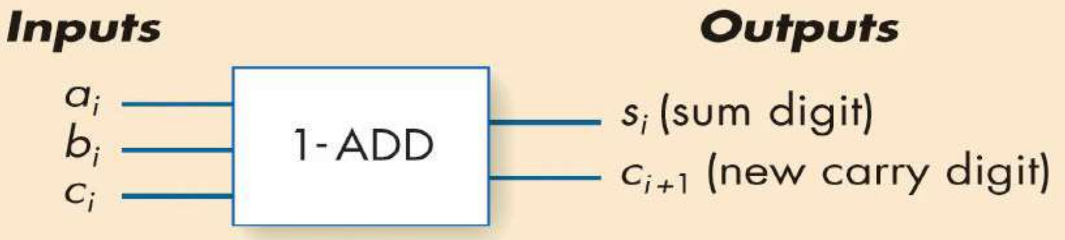
    ((NOT a) AND (NOT b)) OR (a AND b)

An Addition Circuit

- Addition circuit
  - Adds two unsigned binary integers, setting output bits and an overflow
  - Built from 1-bit adders (1-ADD)
  - Starting with rightmost bits, each pair produces
    - A value for that order
    - A carry bit for next place to the left

An Addition Circuit (continued)

- 1-ADD truth table
  - Input
    - One bit from each input integer
    - One carry bit (always zero for rightmost bit)
  - Output
    - One bit for output place value
    - One “carry” bit



Inputs			Outputs	
a_i	b_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figure 4.24

The 1-ADD Circuit and Truth Table

An Addition Circuit (continued)

- Building the full adder
  - Put rightmost bits into 1-ADD, with zero for the input carry
  - Send 1-ADD's output value to output, and put its carry value as input to 1-ADD for next bits to left
  - Repeat process for all bits

Control Circuits

- Do not perform computations
- Choose order of operations or select among data values
- Major types of controls circuits
  - Multiplexors
    - Select one of inputs to send to output
  - Decoders
    - Sends a 1 on one output line, based on what input line indicates

Control Circuits (continued)

- Multiplexor form
  - 2^N regular input lines
  - N selector input lines
  - 1 output line
- Multiplexor purpose
  - Given a code number for some input, selects that input to pass along to its output
  - Used to choose the right input value to send to a computational circuit

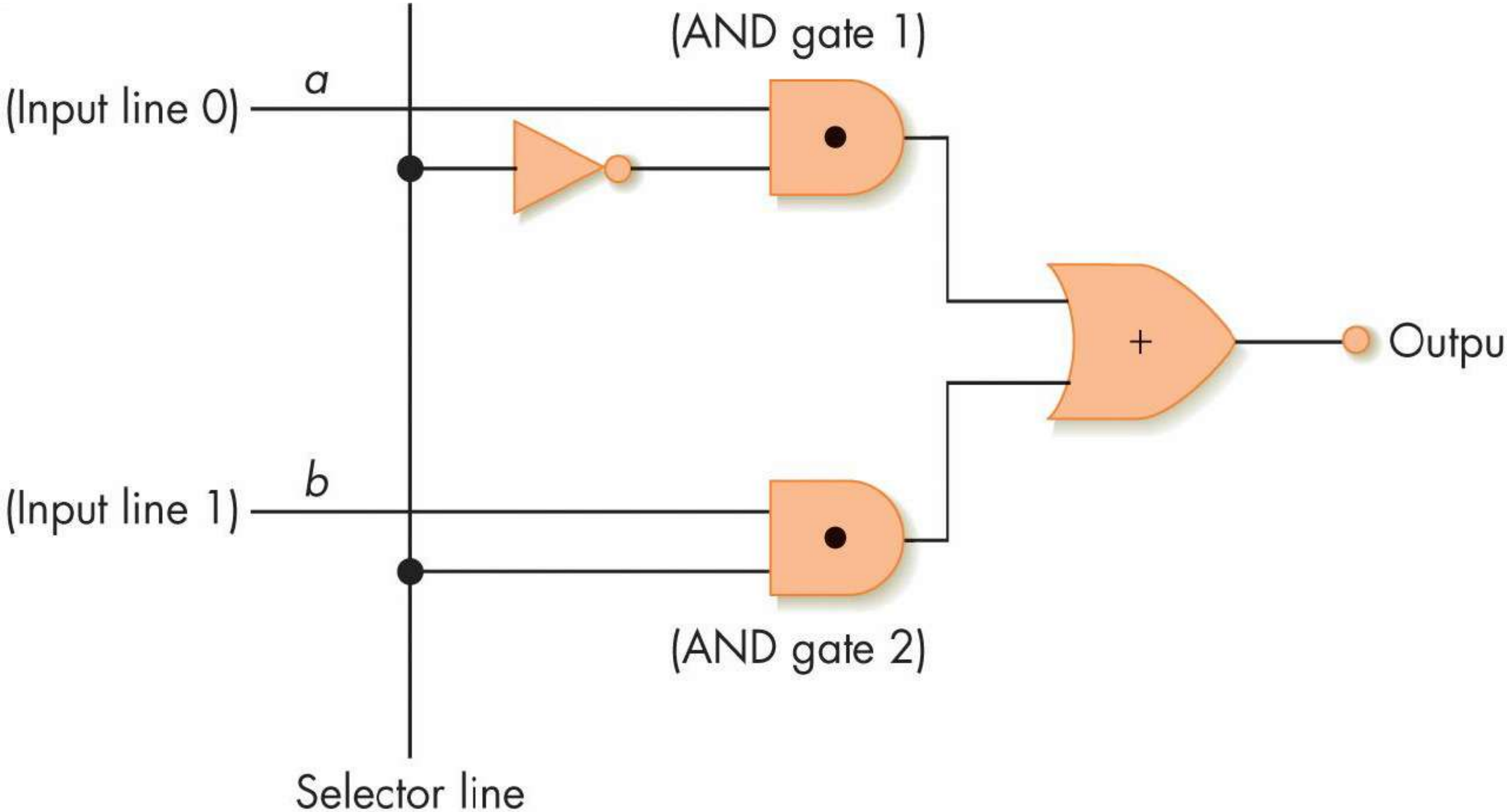


Figure 4.28

A Two-Input Multiplexor Circuit

Control Circuits (continued)

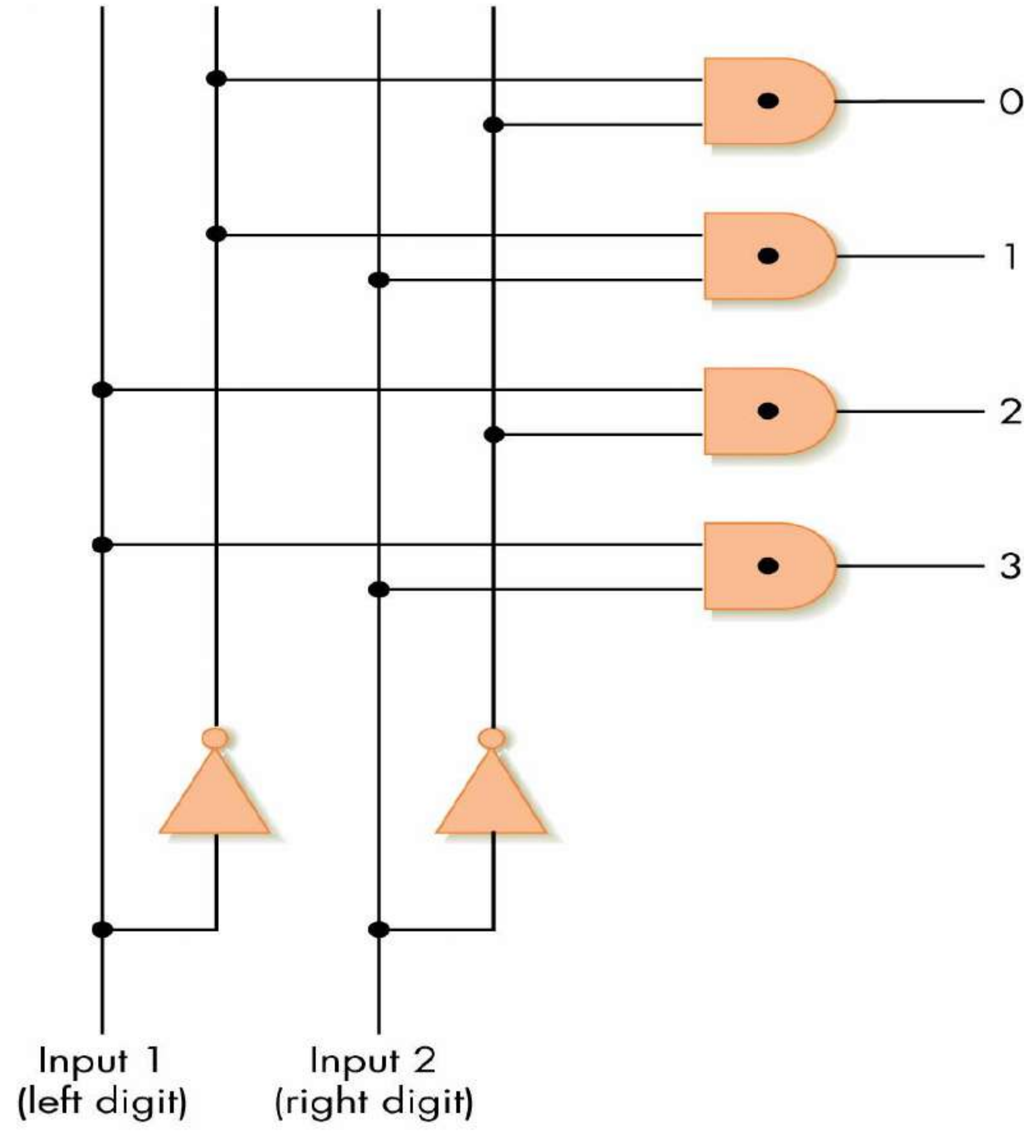
- Decoder
  - Form
    - N input lines
    - 2^N output lines
  - N input lines indicate a binary number, which is used to select one of the output lines
  - Selected output sends a 1, all others send 0

Control Circuits (continued)

- Decoder purpose
  - Given a number code for some operation, trigger just that operation to take place
  - Numbers might be codes for arithmetic: add, subtract, etc.
  - Decoder signals which operation takes place next

Figure 4.29

A 2-to-4 Decoder Circuit



Summary

- Digital computers use binary representations of data: numbers, text, multimedia
- Binary values create a bistable environment, making computers reliable
- Boolean logic maps easily onto electronic hardware
- Circuits are constructed using Boolean expressions as an abstraction
- Computational and control circuits may be built from Boolean gates

BBM105: Week 7, Operating Systems

Slides From: Invitation to Computer Science,Fourth Edition

Objectives

In this chapter, you will learn about

- System software
- Assemblers and assembly language
- Operating systems

Introduction

- Von Neumann computer
  - “Naked machine”
  - Hardware without any helpful user-oriented features
  - Extremely difficult for a human to work with
- An interface between the user and the hardware is needed to make a Von Neumann computer usable

Introduction (continued)

- Tasks of the interface
  - Hide details of the underlying hardware from the user
  - Present information in a way that does not require in-depth knowledge of the internal structure of the system
  - Allow easy user access to the available resources
  - Prevent accidental or intentional damage to hardware, programs, and data

System Software: The Virtual Machine

- System software
  - Acts as an intermediary between users and hardware
  - Creates a virtual environment for the user that hides the actual computer architecture
- Virtual machine (or virtual environment)
  - Set of services and resources created by the system software and seen by the user

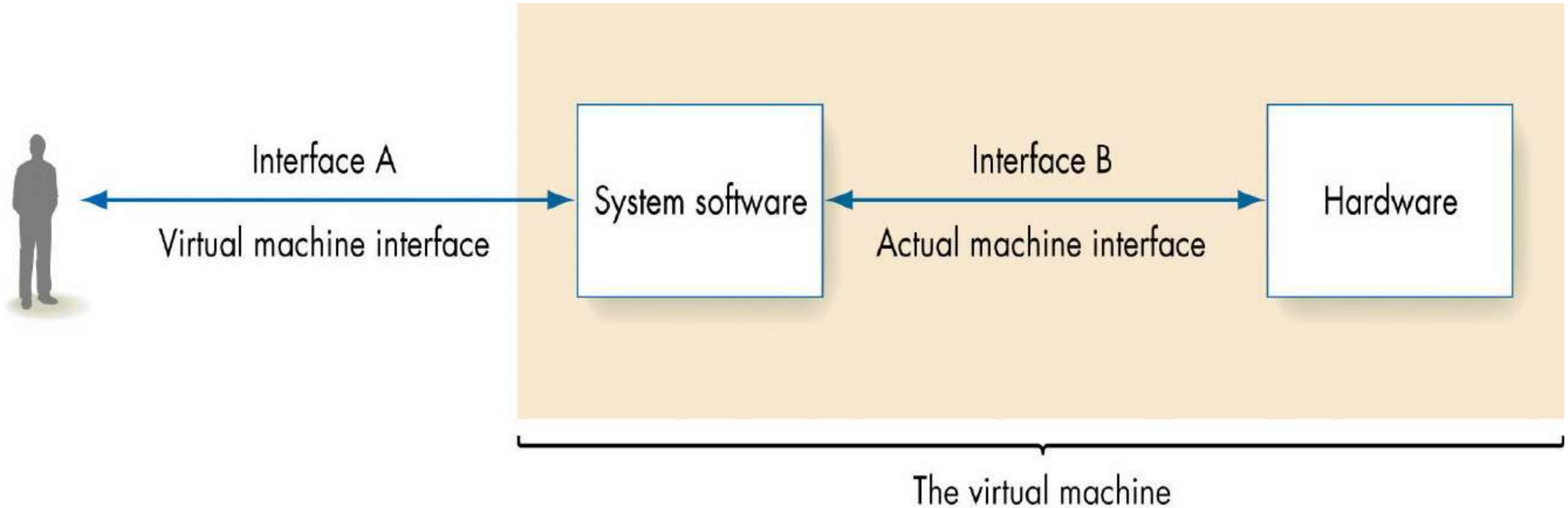


Figure 6.1

The Role of System Software

Types of System Software

- System software is a collection of many different programs
- Operating system
  - Controls the overall operation of the computer
  - Communicates with the user
  - Determines what the user wants
  - Activates system programs, applications packages, or user programs to carry out user requests

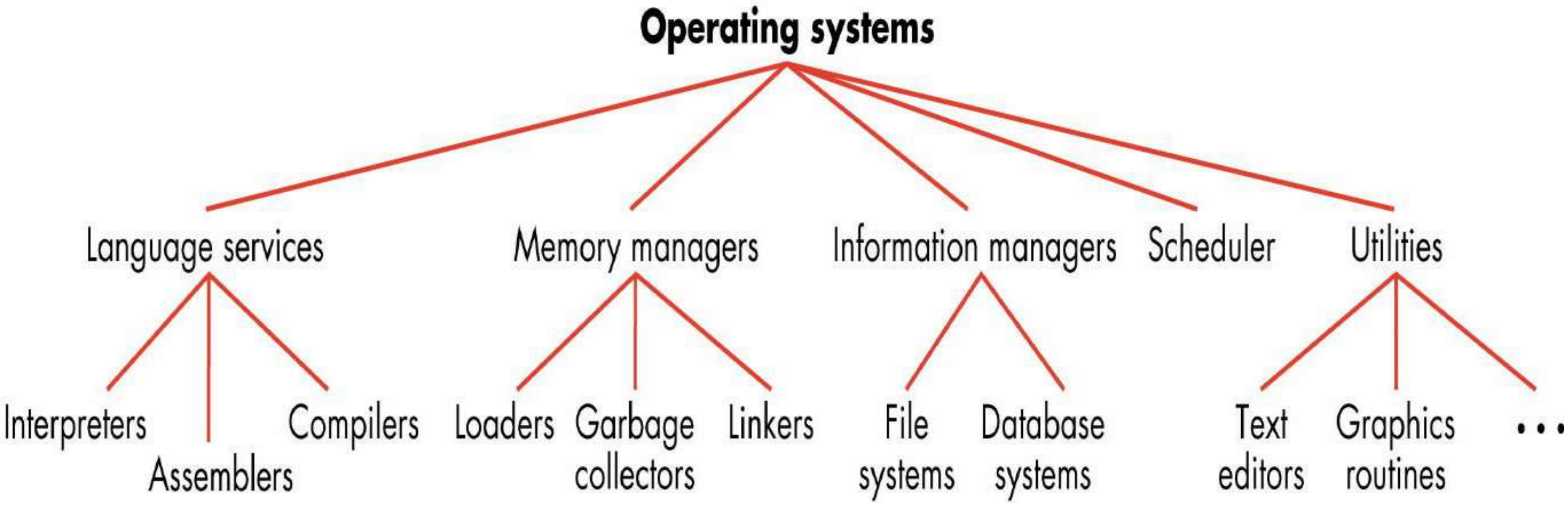


Figure 6.2

Types of System Software

Types of System Software (continued)

- User interface
  - Graphical user interface (GUI) provides graphical control of the capabilities and services of the computer
- Language services
  - Assemblers, compilers, and interpreters
  - Allow you to write programs in a high-level, user-oriented language, and then execute them

Types of System Software (continued)

- Memory managers
  - Allocate and retrieve memory space
- Information managers
  - Handle the organization, storage, and retrieval of information on mass storage devices
- I/O systems
  - Allow the use of different types of input and output devices

Types of System Software (continued)

- Scheduler
  - Keeps a list of programs ready to run and selects the one that will execute next
- Utilities
  - Collections of library routines that provide services either to user or other system routines

Assembly Language

- Machine language
  - Uses binary
  - Allows only numeric memory addresses
  - Difficult to change
  - Difficult to create data

Assembly Language (continued)

- Assembly languages
  - Designed to overcome shortcomings of machine languages
  - Create a more productive, user-oriented environment
  - Earlier termed second-generation languages
  - Now viewed as low-level programming languages

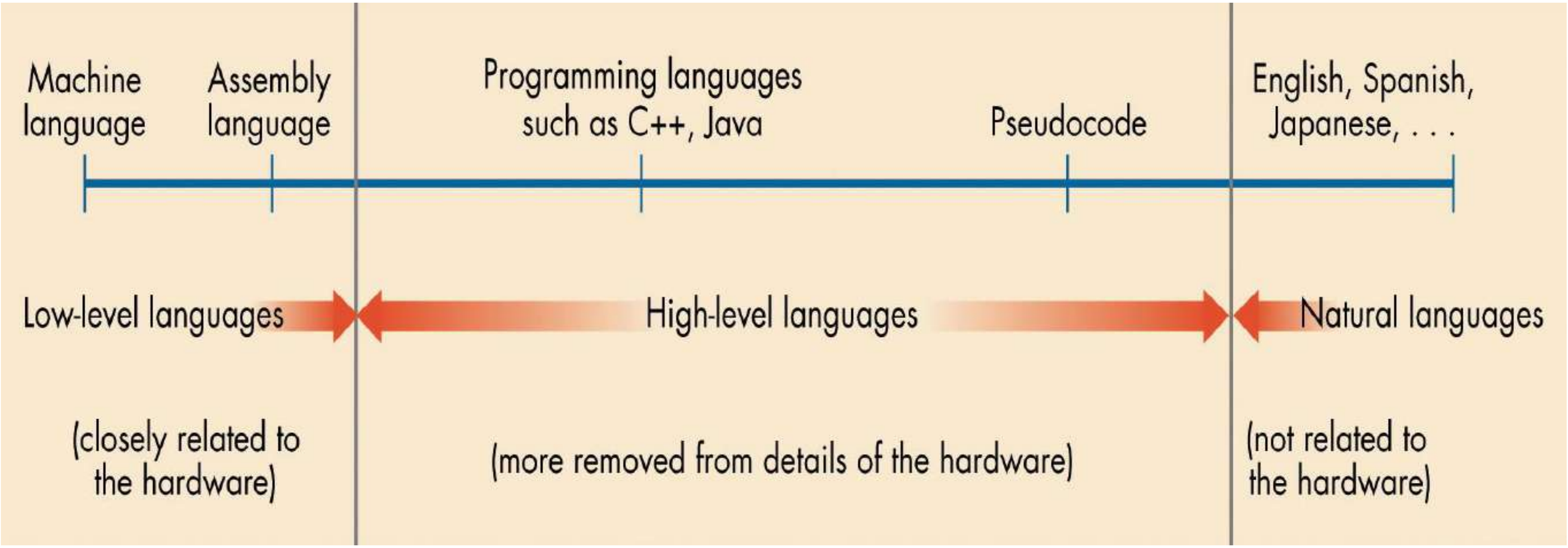


Figure 6.3

The Continuum of Programming Languages

Assembly Language (continued)

- Source program
  - An assembly language program
- Object program
  - A machine language program
- Assembler
  - Translates a source program into a corresponding object program

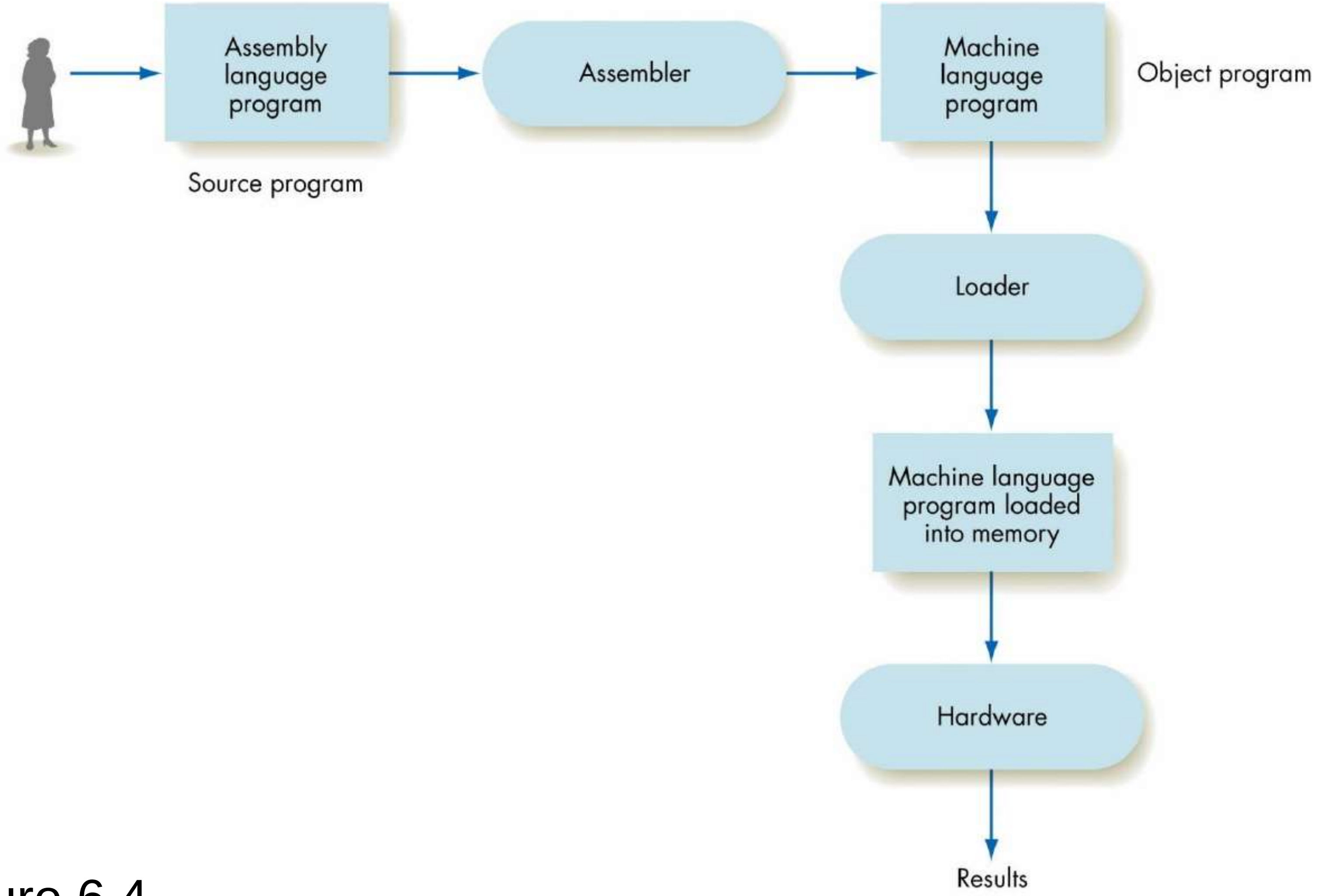


Figure 6.4

The Translation/Loading/Execution Process

Assembly Language (continued)

- Advantages of writing in assembly language rather than machine language
  - Use of symbolic operation codes rather than numeric (binary) ones
  - Use of symbolic memory addresses rather than numeric (binary) ones
- ***We do not go in the details of assemble language!***

Operating Systems

- System commands
  - Carry out services such as translate a program, load a program, run a program
  - Types of system commands
    - Lines of text typed at a terminal
    - Menu items displayed on a screen and selected with a mouse and a button: Point-and-click
  - Examined by the operating system

Functions of an Operating System

- Five most important responsibilities of the operating system
  - User interface management
  - Program scheduling and activation
  - Control of access to system and files
  - Efficient resource allocation
  - Deadlock detection and error detection

The User Interface

- Operating system
  - Waits for a user command
  - If command is legal, activates and schedules the appropriate software package
- User interfaces
  - Text-oriented
  - Graphical

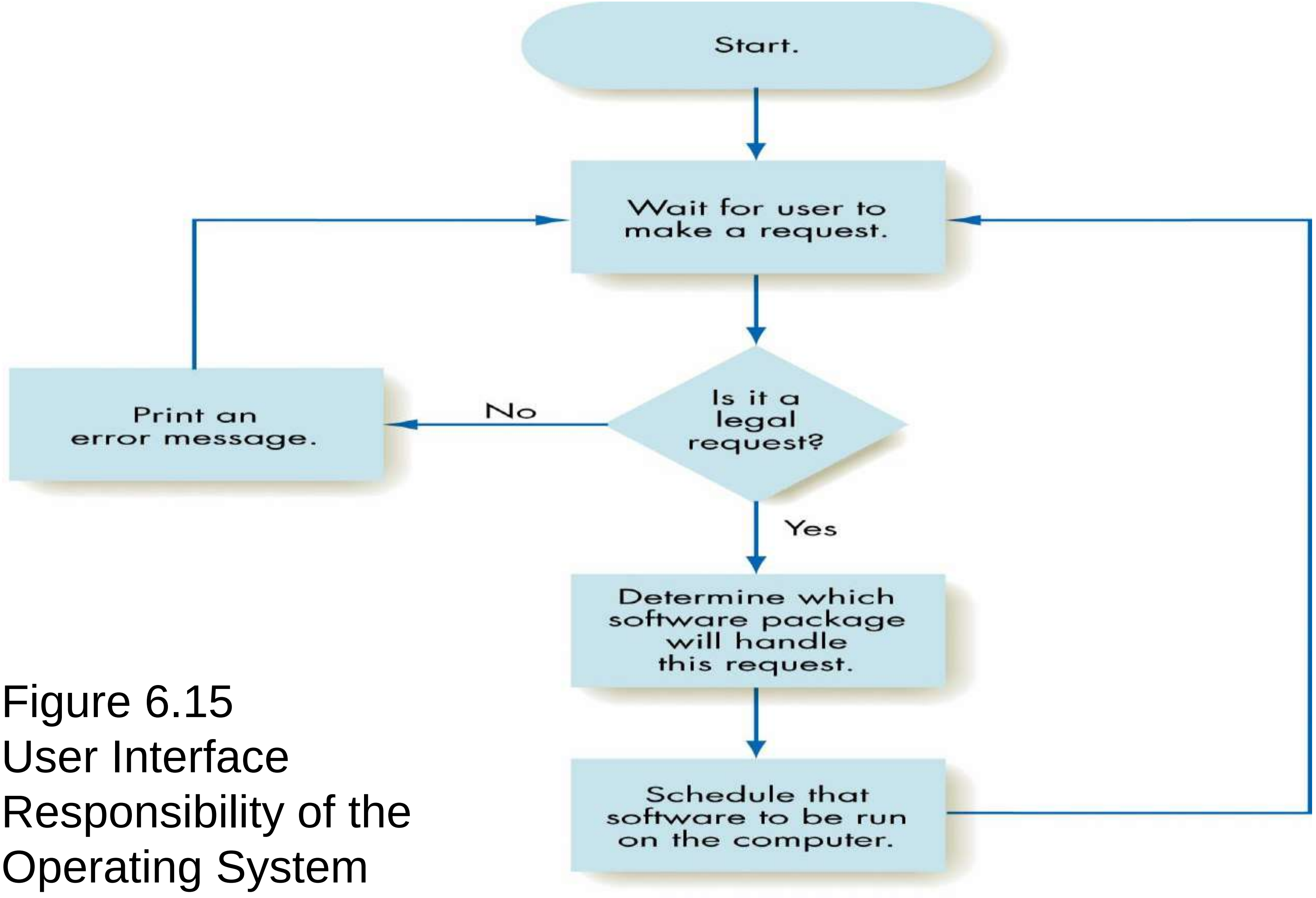


Figure 6.15
User Interface Responsibility of the Operating System

System Security And Protection

- The operating system must prevent
  - Non-authorized people from using the computer
    - User names and passwords
  - Legitimate users from accessing data or programs they are not authorized to access
    - Authorization lists

Efficient Allocation Of Resources

- The operating system ensures that
  - Multiple tasks of the computer can be underway at one time
  - Processor is constantly busy
    - Keeps a queue of programs that are ready to run
    - Whenever processor is idle, picks a job from the queue and assigns it to the processor

The Safe Use Of Resources

- Deadlock
  - Two processes are each holding a resource the other needs
  - Neither process will ever progress
- The operating system must handle deadlocks
  - Deadlock prevention
  - Deadlock recovery

The Future

- Operating systems will continue to evolve
- Possible characteristics of fifth-generation systems
  - Multimedia user interfaces
  - Parallel processing systems
  - Completely distributed computing environments

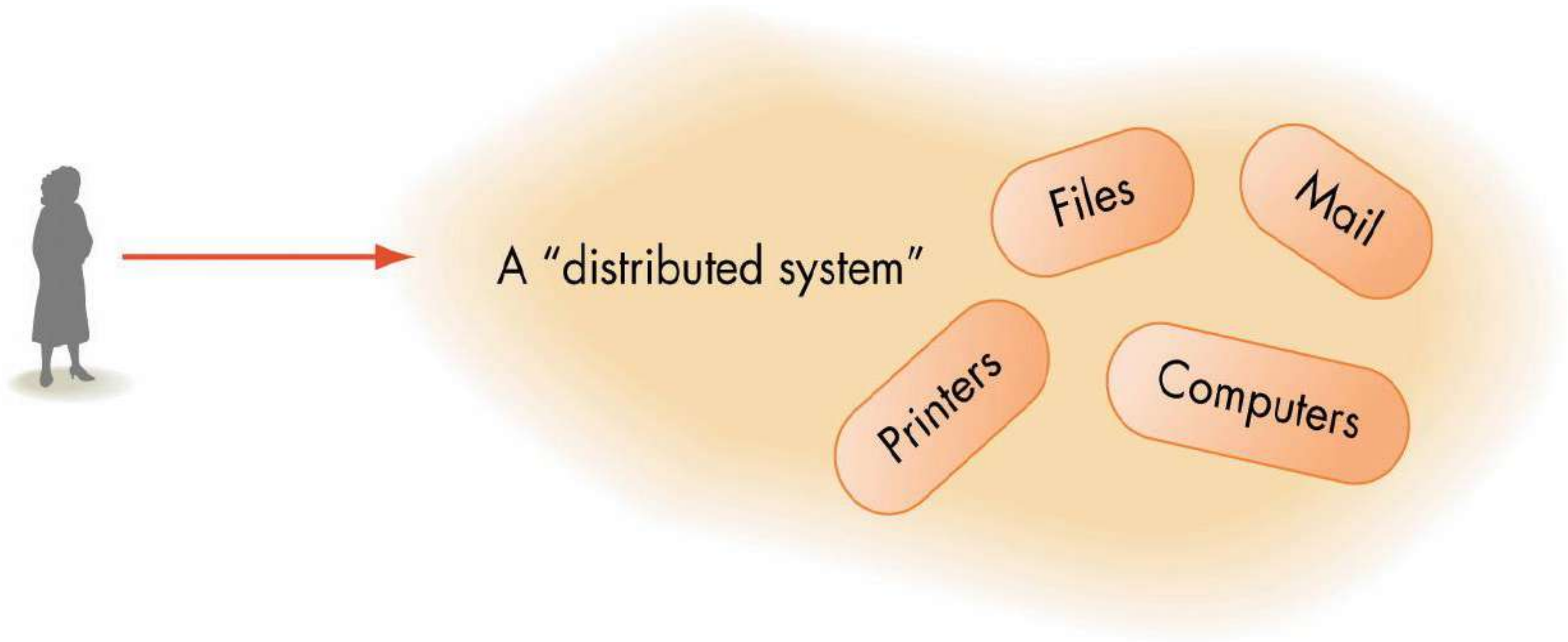


Figure 6.23

Structure of a Distributed System

Generation	Approximate Dates	Major Advances
First	1945–1955	No operating system available Programmers operated the machine themselves
Second	1955–1965	Batch operating systems Improved system utilization Development of the first command language
Third	1965–1985	Multiprogrammed operating systems Time-sharing operating systems Increasing concern for protecting programs from damage by other programs Creation of privileged instructions and user instructions Interactive use of computers Increasing concern for security and access control First personal computer operating systems
Fourth	1985–present	Network operating systems Client-server computing Remote access to resources Graphical user interfaces Real-time operating systems Embedded systems
Fifth	??	Multimedia user interfaces Massively parallel operating systems Distributed computing environments

Figure 6.24
Some of the Major Advances in Operating Systems Development

Summary

- System software acts as an intermediary between the users and the hardware
- Assembly language creates a more productive, user-oriented environment than machine language
- An assembler translates an assembly language program into a machine language program

Summary (continued)

- Responsibilities of the operating system
  - User interface management
  - Program scheduling and activation
  - Control of access to system and files
  - Efficient resource allocation
  - Deadlock detection and error detection