Outline of Computer Science
June 22, 2015

Contents

Note: To look up references, see the Consciousness Bibliography, listing 10,000 books and articles, with full journal and author names, available in text and PDF file formats at http://www.outline-of-knowledge.info/Consciousness_Bibliography/index.html.

MATH>Computer Science

computer science

Mathematical sciences {computer science, mathematics} can be about hardware and software.

computers

Computers input data; run application programs to perform arithmetic, logical, or string operations on input and intermediate data; and output data. Computers are information processors, general symbol manipulators, and efficient universal Turing machines.

Digital computers [Neumann, 1946] use off-on (binary code) switches, in switching networks of digital electronic circuits, to represent data and instructions. Electronic switching devices keep their current state until the next machine step resets switches. [Telephone exchanges are also switching networks, using switches with more than two positions (n-ary code).]

Analog computers use continuous voltage or current magnitudes in electronic circuits to represent numbers, calculate integrals, find slope or gradient, sense coincidences and anti-coincidences, time pulses, shape pulses, combine pulses, amplify pulses, invert pulses, and invert and amplify pulses.

computer parts

The central processing unit (CPU) runs the clock, reads input data from memory, reads program instructions, executes instructions, and writes output data to memories, screens, or printers.

The computer clock sends electric pulses at beginnings of CPU cycles (steps).

Computer memory stores data and instructions.

Input-output devices include tapes, disks, solid-state memories, printers, and screens.

CPUs and memories have registers for holding input values, output values, instructions, results, and register addresses. Information moves in sequence from one register to another, along one serial path or simultaneously on parallel paths.

operating system

Computer operating systems store system configuration, handle input-output operations, manipulate and process files, manage and allocate memory, and begin and end application programs.

Assemblers, compilers, and interpreters change high-level-language programs into assembly language or binary code that can run directly on the CPU.

programs

Computer programs retrieve input, transform information, and store output.

Computer programs are sequences of instructions. A complex instruction set has approximately 200 different instructions:

- Read data (number, symbol, letter, word, date, or boolean) from a memory address into the CPU, or write data from CPU to a memory address.
- Move data from a memory address to a register, or from a register to a memory address.
- Using data retrieved from two registers, perform an arithmetic operation and put the resulting data in a register.
- Using data retrieved from two registers, perform a logical operation (AND or OR) and put TRUE or FALSE in a register.
- Using data retrieved from two registers, perform a comparison operation, to see if numbers are equal, less than, or greater than, or if strings are the same, different, longer, or shorter, and put EQUAL/SAME, MORE, or LESS in a register.
- Depending on a condition (or unconditionally), branch to a new location in a program and perform the instruction there, sometimes returning to the previous location.
- Transform strings or extract substrings from strings.
- Determine if statement A and/or B is (NOT) true, and then perform statement C (conditional): "IF A, THEN C", "IF A AND B, then C", or "IF A OR B, then C".
- If value of i is between m and n, perform x to change i, and then check value of i again (loop): "FOR i FROM m TO n, DO x".
- Execute STOP instruction.

operation

The initial program loader starts the computer system by placing the operating system in memory and starting the central processor unit (CPU).

The operator routine loads the application program.

The data-management program inputs data and controls data flow.

The controller module controls operations, so the CPU executes the first instruction to retrieve, process, or write data.

The CPU continues following the program's instructions to the STOP program instruction. Computers must be able to get to at least one stop instruction and end program operation.

principles

Input data, instructions, intermediate results, and output data together make a state. Turing machines and computer systems have a finite number of states and go from initial state to final state.

Turing machines and computer systems go from state to state deterministically. Current state (instruction, intermediate result, and input) determines next state.

Positive integers can represent states. Turing machines and computer systems make integer series, which operate like algebras and are closed systems.

brains compared to computers

Neuron arrays can be computers [Eccles et al., 1976] [Pellionisz and Llinás, 1982].

Computers and brains have hardware and software: input data, data reading, computer memory, information processing, central processing unit, operating system, application programs, data writing, and output.

Brains and computers are integrated systems that store instructions, input, intermediate results, and output; read input and write output; process information using instructions, rules, and programs; and have states.

Brains combine features of analog and digital computers, because they have switches, clocks, continuous signals, thresholds, and multiple connections.

Brains combine features of serial and parallel computers, because many neurons simultaneously and continually read and write data according to instructions to many neurons.

Brains and computers have registers and memories that hold data, which can change.

Brains and computers have a central processing unit and operating system to run instructions and execute processes.

MATH>Computer Science>Hardware

switching network

Computers are switching networks {switching network}|, like telephone exchanges. CPUs are electronic switching devices, which keep current state until next machine step resets switches. Instructions, from registers, open and close switches that activate hardware.

MATH>Computer Science>Hardware>Kinds

analog computer

Computers {analog computer}| can have electronic circuits that use electrical or physical magnitudes, rather than digital codes, to represent numbers and manipulate physical quantities algebraically, rather than digitally.

types

Analog computers {integrator computer} {summing integrator} can calculate integrals. Differential analyzers find slope or gradient. Analog computers can sense coincidences, sense anti-coincidences, combine pulses, amplify {summing amplifier}, time, shape, invert pulses {inverter}, invert and amplify {inverter amplifier}, find voltages {coefficient potentiometer}, supply constant voltage {constant-voltage supply}, and amplify selectively {operational amplifier}.

DNA computer

DNA can code combinations {DNA computer}. Filters can look for answer pieces. DNA hybridization indicates solution. DNA performs faster than computers.

MATH>Computer Science>Hardware>Kinds>Size

server

Large computers {server, computer} can receive input from many smaller computers {client}.

mainframe computer
Bigger computers {mainframe computer}| are faster, have more memory, and have more input/output terminals.

minicomputer
Midsize computers {minicomputer}| can perform most mainframe-computer functions but slower and cheaper, with less memory and less input/output terminals.

workstation
Smaller computers {workstation}| can function independently and perform complex and fast calculations.

microcomputer
The smallest computers {microcomputer}| function independently and perform most functions.

MATH>Computer Science>Hardware>Parts

computer parts
Computer systems have physical parts {computer, parts} {peripheral}. CPU reads instructions and data and calculates. Memory stores data and instructions. Input receives data. Output prints, displays, or sends data to another device.
relations
Input or output devices connect to peripheral-control bus, which connects to input-output processor, which connects to CPU and memory. Central processor connects to input-output processor and memory. Memory connects to input-output processor and CPU.
computer clock
Computer clocks send electric pulse at beginning of CPU cycles or steps. Time pulse into AND-circuit flip-flop sets flip-flop circuit. Time-pulse length must be long enough to set flip-flop. Signals go from one flip-flop circuit to the next in one time pulse. Time interval between CPU clock pulses must be long enough to let signal stabilize and to let pulse pass through longest network.
data protection
Duplicate equipment can preserve data. Comparing twin equipment can check for errors. Duplicate equipment can share work. One can switch on as backup, if other fails. Backup methods {failsoft} can limit data loss from failures. Central-control unit can switch off parts that have failed or are being serviced {controlled reconfiguration}.

central processing unit
Computers have a control part {central processing unit}| {central processor} (CPU).
functions
CPUs route signals, control input and output, and decode and execute instructions.
modules
CPUs have hardware modules that receive one or two inputs, perform functions, and send results to memory or output. Modules count {counter circuit}, add {adder circuit}, make one signal from many signals {decoder}, make several inputs into code {encoder}, or make one code into another code {translator circuit}. Gates can perform logical functions. Comparator compares number or string values.
subtraction
Subtraction adds input-number 10s complement to first number, using adder circuits.
multiplication
Multiplying repeats adding and shifts register using adder circuit.
division
Division has carry detector and restoring method, using adder circuit.

register
CPUs and memories have locations {register, computer}| for holding input values, output values, instructions, results, and addresses. Registers {look-ahead register} can hold next instruction or input value. Registers {accumulator register} can hold processing result or point to memory location holding result. Registers {address register} can hold other-register locations. Registers {operation code register} can hold operation instructions. Registers {temporary storage register} can hold data to transfer to other registers. Registers {stack} can hold value sequences. Stacks {stack counter register} can hold word address.

Register series {multi-register} can have fixed addresses for registers. Register series {stack processor} can be in CPU memory, so operations use only pushdown and popup in stack, with no addresses.

transfer

Information can move in sequence from one register to another on one path {bit serial} {character serial}. Information can move on parallel paths simultaneously {bit parallel} {character parallel}.

MATH>Computer Science>Hardware>Circuit

differentiator

Electrical circuits {differentiator}| can find difference between two inputs. Differentiators can have input voltage from ground through capacitance to output voltage node, which connects to ground using diode or resistor. RC-circuit time constant must be small.

electron sink

Wires can connect device to ground {electron sink}| {electrical ground}, which can absorb any number of electrons.

electron source

Circuit currents come from batteries or other voltage sources {electron source}|.

flip-flop circuit

Electrical circuits {flip-flop circuit}| can change output voltage from zero to unit voltage and from unit voltage to zero, after input. Flip-flop circuits can have two inputs, each from transistor ground to base. For both transistors, emitters connect to ground. For both transistors, collector connects to other-transistor base. Both transistors have grounded emitters. Second-transistor collector connects through resistance to output voltage node, which connects through capacitor to ground.

hold-on relay

Electrical circuits {hold-on relay}| can maintain voltage until switch changes. Hold-on relays can have grounded inductor, which connects to two switches. First switch leads to third switch. Second switch leads to third switch but also connects to ground through capacitor. Output voltage is across third switch. Output voltage is 0 or 1 and does not change until one of first two switches changes. Hold-on relay in OR circuit has output voltage 1 if OR is true. Hold-on relay in AND circuit has output voltage 0 if AND is true.

integrator circuit

Electrical circuits {integrator circuit}| can add two inputs. Integrators can have input voltage from ground through resistor to output voltage node, which connects through capacitor to ground. RC-circuit time constant must be large.

MATH>Computer Science>Hardware>Circuit>Gate

logic circuit

Electrical circuits {logic circuit}| {logic gate} can perform logical operations, such as AND, OR, NOT, NOT OR {NOR gate, circuit}, and NOT AND. For NAND gates, P|Q = NOT(P AND Q). For AND gates, if all inputs are 1, output is 1. For OR gates, if at least one input is 1, output is 1. For inverter gates, if input is 1, output is 0. NAND gates are AND, followed by inverter. NOR gates are OR, followed by inverter [Church, 1956].

AND gate

Logic circuits {AND gate}| can be equivalent to (A v B) or (A and B). AND gates can have two input voltages, each from ground to reverse diode. Output currents combine at node whose output voltage is across resistor and capacitor, each of which is parallel to input diode. AND gates can have two input voltages, each from ground through resistor to transistor base, which also connects through resistor to ground. One emitter connects to ground, and other emitter connects to first-transistor collector. Output voltage is at second collector, which lies across resistor and capacitor to grounded emitter.

AND-OR gate

Logic circuits {AND/OR gate} {AND-OR gate}| can have two system inputs, controller input, and output. Output of 2 or 3 sends output. If controller is 0, circuit is AND gate. If controller is 1, circuit is OR gate.

NEGATIVE AND gate

Logic circuits {NEGATIVE AND gate}| {NAND gate} {Sheffer stroke} can be equivalent to ~(A v B) or not (A and B). NEGATIVE AND gates can have two input voltages, each from ground to diode, whose output currents combine at node whose output voltage is across resistor and capacitor to ground. NEGATIVE AND gates can have two input voltages, each from ground to reverse diode, whose output currents combine at node, which connects across resistor to ground and connects to diode. Diode output current goes to transistor base, which connects across resistor to ground. Emitter connects to ground. Output voltage is from collector to grounded emitter, across resistor.

NEGATIVE OR gate

Logic circuits {NEGATIVE OR gate}| {NOR gate, negative or} can be equivalent to ~(A ^ B) or not (A or B). NEGATIVE OR gates can have two input voltages, each from ground to reverse diode, whose output currents combine at node. Node connects across resistor to ground and leads to diode, whose output current goes to transistor base, which connects across resistor to ground. Emitter connects to ground. Output voltage is at collector, which connects across resistor to ground.

NOT gate

Logic circuits {NOT gate}| {inverter circuit} can be equivalent to ~A or not A. NOT gates can have input voltage from ground through resistor to transistor base, which lies across resistor and capacitor to ground. Emitter connects to ground. Output voltage is across capacitor to collector, which connects across diodes and resistors to ground.

OR gate

Logic circuits {OR gate}| can be equivalent to (A ^ B) or (A OR B). OR gates can have two input voltages, each from ground to diode. Two output currents combine at node whose output voltage is across resistor to ground. OR gates can have two input voltages, each from ground to diode, whose output currents combine at transistor base. Base also connects through resistor and capacitor to ground. Emitter connects to ground. Output voltage is at collector, which connects across resistor and capacitor to grounded emitter.

MATH>Computer Science>Hardware>Data Conversion

code-decoder

Analog signal magnitude can convert to digital and then reconvert for display {compression-decompression} {code-decoder} {codec}, possibly using time-division multiplexing.

pulse-code modulation

Sampling analog signal magnitude at regular intervals can code binary signals {pulse-code modulation} {analog-to-digital}, possibly using multiplexing {time-division multiplexing}.

pulse-width modulation

Digital input can go through low-pass filter with frequency close to digital frequency {pulse-width modulation} {digital-to-analog}. Negative feedback in one-bit digital-to-analog converters {Digital to Analog Converter} (DAC) can create high-pass filter that eliminates noise {pulse-density modulation} {oversampling DAC}. Digital bits can have resistances across voltage or have current sources {binary weighted DAC} {R2R ladder DAC} {segmented DAC} {hybrid DAC}.

MATH>Computer Science>Hardware>Memory

memory of computer

Input or output devices {memory, computer} store data on magnetic disk {tanking} or in semiconductor memory {scratch pad memory}. Memories have byte sets. Memory-control devices assign addresses to bytes and read, write, sense, or inhibit bits.

cache memory

Temporary memories {cache, memory}| can hold data.

holographic memory

Memories {holographic memory} can use interference patterns to store information and coherent input to retrieve information.

magnetic bubble memory
At surface positions, current can change magnetized-bubble direction {magnetic bubble memory}.

magnetic core memory
Magnetic cores can be registers {magnetic core memory}. Wire circles carry current to make solenoids with magnetism. Wires through magnetic cores can turn current off or on. Other wires through magnetic cores can read whether magnetism is present or not.

random-access memory
Memory controllers can read or write disks {random-access memory} (RAM) at any register location at all times. System stores information at random locations. For strings, logically add bits to make unique random numbers. Place string in memory at random-number position. Add line to table {hash table} relating memory-location random numbers to strings. To recover string from memory, look up random number in table. At that location, retrieve string from memory. Logically subtract bits from string.

serial memory
Tapes {serial memory} read and write in sequence.

thin-film memory
Disk or tape magnetic films {thin-film memory} can magnetize or not at all positions.

MATH>Computer Science>Hardware>Input-Output

input of computer
Programs obtain data values or programming instructions {input, computer} by opening and reading files. Input devices include magnetic tape, magnetic disk, magnetic drum, card reader, microfilm, microfiche, keytape from typewriter to tape, paper-tape reader, mark-sense reader, optical character recognition {optical character recognition} (OCR), cathode ray terminal, data cells, magnetic card, diskette, floppy disk, and flash memory card.

output of computer
Programs display, print, send data to other devices, or make data-value or programming-parameter files {output, computer}. Output devices include de-collator, burster, folder, line printer, laser printer, card puncher, paper tape puncher, key-to-disc, and key-to-drum.

file transfer protocol
File transfer protocol {file transfer protocol} (FTP) allows public file access via anonymous log on. There is a user account for anonymous logons (GUEST by default) and a default home directory.

channel of computer
CPUs can connect {channel} to one input or output device {fixed channel} or switch from one input or output device to another {floating channel}.

line of computer
Input-output lines {line, input and output} {input line} {output line} connect CPU and device. Direct lines {bit stream} can connect one input or output device to CPU. Leased telephone lines and modems connect one input or output device to CPU. Multiple lines connect to switching network, which connects to CPU. CPU lines {multi-drop line} can have several modems, so CPU polls and selects data. CPU lines {line concentrator} can receive direct and telephone lines, poll them, and send bit-stream packets to CPU.

MATH>Computer Science>Hardware>Input-Output>Devices

adapter of computer
Computer devices {adapter} can make bits into line characters.

data communications controller

Computer devices {data communications controller} {datacom} can control data input and output from different terminals.

types

Datacoms {uniplexor} {single line control} (SLC) can take bits from one line and pass characters to CPU. Datacoms {multiplexor} {multi-line control} (MLC) can take bits from many lines, sort bits by logic, and send characters to CPU. Datacoms {front-end processor} (FEP) can take bits from several lines, assemble them into words, and send words to CPU. Datacoms {data communications processor} can take bits from one line, assemble them into words, add message header, queue messages, and send them to memory or input-output processor.

modem

Devices {modem}| {data set} can convert digital bits to analog data, or vice versa, or alter line bit streams.

sort-merge of computer

Devices {sort-merge} can manipulate input and output forms.

MATH>Computer Science>Hardware>Robot

robot machine

Robots {robot, computer} can recognize patterns [Aleksander, 1983] [Liang et al., 1997]. QRio android robot can move and converse.

Kismet is a robot head whose parts can move like human head parts [Breazeal, 2001]. If it is too close to see, it cranes back, and if it is too far, it cranes forward. It checks for movement, skin color, and saturated colors and looks in weighted direction. State depends on happiness, stimulation, and willingness for new stimuli. State affects where it looks. It checks pitch for approving, disapproving, attending, and soothing patterns, as used worldwide by mothers to babies, but it has no language ability. It can make sounds in pitch patterns. It can move eyebrows, lips, and ears to reflect happiness, stimulation, and willingness for new stimuli.

silicon retina

In neuromorphic systems {silicon retina}, photodetectors output voltages proportional to current logarithm. Resistive network locally weights averages, so point effects decrease exponentially with distance. Retinal output is difference between local and overall voltage [Mead and Mahowald, 1991].

MATH>Computer Science>Software

software

Application programs {software} run on hardware.

bottleneck in process

Where inventory piles up indicates where processes slow down {bottleneck, process}. To remove bottlenecks, use smaller batches, decrease flow rate, improve efficiency and productivity there, increase people or tools there, build an alternate pathway, and/or modify process to eliminate bottleneck. Typical design weaknesses include confusing hierarchy of screens, incomplete instructions or information, poor screen design and layout, and/or unjustified assumptions about user knowledge.

contextual design

Contextual design {contextual design} finds what customers need and do and how the organization develops, checks, and sells its products, using focus groups, protocol capture, and prototyping.

Contextual design diagrams the roles of people and things in processes: Information and materials flow among people and objects. People assume information, have organizational relationships, and interaction types. Whole organization and its subgroups perform tasks. Physical objects have assigned descriptions and locations.

Organization users have goals, requirements to reach goals, and bases for satisfactory goal completion. Users have inputs with sources, processes, and outputs with destinations. Users have physical, business, and information contexts. Users have information, skills, and mental models, and talk to themselves while doing tasks.

Customers use products in specific ways, have needs and priorities, and have specific interactions with organization objects, information, and people.

Information and material flows, people relationships, people tasks, and physical objects have relations and fall into groups by time, space, task, flow, or culture.

People have roles, and objects are parts of processes, and role-and-process groups model the whole organization.

Contextual design fits changes and new roles and processes into the organization model at the appropriate location to improve functions but minimize how much people must learn.

focus group

Programmers can observe 6 to 8 users {focus group} to see how they solve a problem using software. Focus groups help find the basis of problems, not just symptoms and solutions.

pilot study

small-scale studies {pilot study}.

protocol capture

Programmers can find the protocol {protocol capture} an average user and/or an expert uses to do a (complex) process, before building software or with built software. Story boards can show processes.

prototyping

Programmers can roughly design papers or screens {prototyping} for a problem, to test and modify quickly, to build user interfaces.

trigger event

Events {trigger, event} can cause users to begin searches or other actions.

usability

Usability {usability} involves information, objects, and results. People and computers use data, object and part definitions, and instructions about procedures. Objects receive information from an object, transform received information into new sendable information, or send information to another object, and all three steps are a work task. Results are goals of a person, achievable by processing information in the system and the user.

Work flows have bottlenecks, where processes slow and inventory piles up. Businesses concentrate resources on improving bottlenecks, typically user interface (UI) design of tools and menus. Tasks must be relevant to goals and relatively easy to perform and understand.

Work items are analyzed for relevance, speed and ease of search and transfer, ease of understanding, and possibilities of mistakes. Users perform relevant tasks with more attention and good will. Understanding is better if all terms defined, steps read easily, necessary preconditions noted, branches noted, procedure accurate and tested, and steps presented one at a time. Speed is better if fewer steps, less data, fewer branches, fewer decisions, and fewer clicks.

Businesses typically reallocate work items from human objects to inanimate objects to reduce information burdens on users. Systems can guide users to information by directed search, using context-sensitive help and dialog-box help. Systems display only the most needed information. Distinctive locators include pop-up help, labels, and grouped menu items and interface objects. Wizards and coaches present a defined way of working.

usability testing

Test usability on typical user tasks by focus groups and protocol capture. Identify users' goals, current knowledge, and steps of learning to use product. Qualify users as new or existing and by knowledge level. Measure task times, successes, and errors, and user fatigue, confusion, frustration, and satisfaction.

usability engineering

Usability engineering {usability engineering} solves usability problems. Work flows have bottlenecks, where processes slow and inventory piles up. Usability engineering concentrates resources on improving bottlenecks, typically user interface (UI) design of tools and menus. Tasks must be relevant to goals and relatively easy to perform and understand. Work items are analyzed for relevance, speed and ease of search and transfer, ease of understanding, and possibilities of mistakes. Users perform relevant tasks with more attention and good will. Understanding is better if all terms defined, steps read easily, necessary preconditions noted, branches noted, procedure accurate and tested, and steps presented one at a time. Speed is better if fewer steps, less data, fewer branches, fewer decisions, and fewer clicks. Usability engineering typically reallocates work items from human objects to inanimate objects to reduce information burdens on users. Systems can guide users to information by directed search, using context-sensitive help and dialog-box help. Systems display only the most needed information. Distinctive locators include pop-up help, labels, and grouped menu items and interface objects. Wizards and coaches present a defined way of working.

work flow

Users have a work flow {work flow} they follow and a mental model of what they are supposed to do (goals) and how they do it (protocol). Users have skills and experience that contribute to performance. Users have a work environment that constrains their performance.

MATH>Computer Science>Software>Coding

coding of data

Storing or transmitting information can use coded symbols {coding, computer} {encoding, computer} {decoding}.

types

Codes can use binary numbers, decimal numbers, words, images, and switches. Code size depends on possible-symbol number and probability.

probability

Symbols can have equal or unequal probability. Binary code use symbols 0 and 1, with equal probability. Probability is base-2 logarithm: P = - log(2^1) = 1/2 for one position. Information {average unexpectedness} is greatest if all symbols are equally probable.

information

Information measurement uses binary units. For binary code, with two equal-probability symbols, information in one position is 1 bit, 2^1 = 2. Information in three positions is 3 bits, 2^3 = 8, with eight possible series, such as 010.

MATH>Computer Science>Software>Coding>String

block of data

Machines store and transmit data in record sets {block, data}| containing 2000, 4000, or 8000 bytes in fixed-length or variable-length blocks. Longer data blocks have fewer recombinations and can process faster.

packet of data

Message or information packets {packet, information} {information packet} {message, information} are only for recipients that can decode them, using same methods as for encoding. Communication requires similarities between senders and receivers. Messages must have marks to show that they are messages. Messages reflect coding method.

MATH>Computer Science>Software>Coding>Methods

block switching

If several inputs or outputs are waiting for transmission, CPU sends blocks in the most-efficient way {message switching} {block switching}.

Data Encryption Standard

Binary messages can split into sections and recombine many times, using methods {Data Encryption Standard} (DES) encoded in 56-bit keys. Methods can use three different keys {triple DES}.

multiplexing of data

Data can separate into small chunks, travel in pieces through different channels, and then reassemble {multiplexing, data}|.

packet switching

Message can split into parts that take alternate routes and then recombine at delivery {packet switching}|. In modems, packets are strings of 8-bit bytes, each with start and stop bit.

packing of data

Decimal-number input can convert to binary numbers {packing, data} {data, packing}. Binary numbers can convert to decimal-number output {unpacking}.

ultrawideband

Data can travel on carrier frequencies by AM or FM, or data can be in pulses {ultrawideband}| (UWB).

MATH>Computer Science>Software>Coding>Codes

ANSI code

Codes {ANSI code} can use 7, 8, or 9 bits to represent 128 or 256 printable and unprintable characters and symbols. ANSI is similar to ASCII.

ASCII code

Codes {ASCII code} can use 7, 8, or 9 bits to represent 128 or 256 printable and unprintable characters and symbols.

binary code

Memory locations or switches can have two states {binary code}: 0 or 1, on or off, zero voltage or unit voltage.

Manchester encoding

0 is impulse-pause, 1 is pause-impulse, and start or stop is impulse-impulse {Manchester encoding}. Start-bit impulse interval sets transmission rate, so transmission rates can vary.

MATH>Computer Science>Software>Data Hierarchy

data hierarchy

Data is in bits that make bytes that make words that make records that make files that make directories {data hierarchy}.

byte

Bit sets {byte}| can hold one binary-coded character or symbol, such as letters, numbers, and punctuation marks. Eight-bit bytes can code 2^8 = 256 different symbols. Bytes can have 8, 16, 32, or 64 bits. Eight-bit bytes can have two four-bit parts {nibble}. Nibbles code numbers from 0 to 15 in hexadecimal.

word of data

Byte sets {word, data}| can hold values or instructions. Word length can be constant, for more speed. Word length can be variable, for more flexibility and memory space. Words can have extra bits {tag bit} {word tag} to indicate data, code, control instruction, descriptor, date, or read-only. Words can have extra bits {correction bit} for error correction. Error-checking codes correct one-bit errors automatically and flag two-bit errors for automatic repeat.

field of record

Sets {field, record} of one or more words can hold strings or numbers. Fields can hold phrases, sentences, and instructions {data element}.

record of data

In files, field sets {record, data}| can hold data lines. Records typically have 80, 256, 2048, or 4096 bytes. Records have addresses. Records can have next-record address {pointer}. Using pointers, record series {thread} can link. Records {multi-threaded element} can be in more than one logical file.

list of records

Records can have pointers to next-record address, linking records {list, computing}. Records can have indexes. List elements can point to positions in address table {inverted list}. Last list record can point to first record {closed list} {ring, record}. Records can have pointers in both directions. Hierarchies and networks have related data elements. Memories can be lists, with nodes and node links.

file of data

Tables or record arrays {file, computing}| can store programs and/or data values. Files are record sets {data collection}.
parts
Files have initial information {file header, file} and end markers {end-of-file marker} (EOF).
file operations
Sort file lines in alphabetical, numerical, or other order. Compare files line-by-line, for differences or similarities. Delete (Remove) file from directory. Save file in directory. Copy saved file to another directory or to same directory with new name. Rename file.
reading
Computers can read files bit by bit in sequence from start to finish {serial file} or at any point {random access file}. Files can have no index {direct file}, sequential index {index sequential file}, or random index {index random file}.
database
Computer files {master file} can hold reference data or database. Files {transaction file} can contain new or temporary data to update master file.
output
Files {output file} can result from processing.

file access
Computers open and close files to read or write data {file, access}. Reading and writing can be from and to text files in linear order {sequential access}, such as appending to files. After opening sequential-access files for writing output or appending data, computers must close files before opening files for reading input. Reading and writing can be from and to records {random access}, which have addresses, such as in databases. After opening random-access files, computers can write to files and then read record without closing files first. Reading and writing can be from and to bitmaps {binary access}, which have byte positions. After opening binary-access files, computers can write to files and then read record without closing files first.

directory of computer
Named file groups {directory, computer}| form hierarchies from root directory to leaf directories. List or show directory files by alphabetical order, date saved or last used, or size. Display current directory name. Change to another directory. Make directory in higher directory. Remove directory from higher directory.

MATH>Computer Science>Software>Data Hierarchy>File

Comma-Separated Value file
Text files {Comma-Separated Value file} (CSV) can contain record lists, with record values separated by commas. Values are in quotes. Records end with newline character.

DOC file
Microsoft(R) Word, WordPerfect (TM), and similar word-processing programs save documents as binary files {DOC file} with DOC extension. Word-processing programs can convert DOC-file formats. In Microsoft Word, files can have Table of Contents (TOC) and Index, as well as links, fields, and hidden text.
contents
Heading paragraph styles determine TOC paragraph styles and indentation levels. TOC page numbers link to headings.
index
Text can have index fields, which you can hide or show. In Microsoft Word, field has format: special-left-brace_space_XE_space_phrase_space_special-right-brace. To search for fields, use ^d. To insert fields, use Ctrl-F9 or Insert menu and Field command. Page numbers in Index are not links.

Framemaker file
Adobe Framemaker (TM) word-processing program uses a proprietary document format {Framemaker file} with extension FM. You can import files from other formats and export Framemaker files to other formats. Other applications typically cannot open Framemaker files. Adobe Acrobat (TM) can open Framemaker files.

HTML file

Hypertext Markup Language (HTML) is an example of Standard Generalized Markup Language (SGML). Such files (SGML file) {HTML file} are text files.

tags

HTML uses tags to open and close formatting. Tags can be in any order or relation. Tags start with less-than signs and close with greater-than signs.

semantics

HTML has no semantics.

comment

<!-- comment --> inserts comment.

format

HTML files have a document-type line first, heading section second, and body section last.

document type line

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"> starts HTML files.

beginning and ending lines

<HTML LANG="en-US"> is after the document-type line. </HTML> is at file end.

heading block

<HEAD> is after <HTML>, to start heading block.

<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=UTF-8"> names content type for search engines.

<META HTTP-EQUIV="Expires" CONTENT="0"> expires page for search engines.

<TITLE>Title</TITLE> inserts title.

<META HTTP-EQUIV="keywords" CONTENT="keyword1, ..., keyword25"> or <META NAME="keywords" CONTENT="keyword1, ..., keyword25"> specifies up to 25 keywords for search engines.

<META NAME="description" CONTENT="25-word description"> describes HTML pages for search engines, using up to 25 words.

<META NAME="robots" CONTENT="noindex"> excludes indexing by search engines. <META NAME="robots" CONTENT="nofollow"> excludes searching for links by search engines. <META NAME="robots" CONTENT="noindex, nofollow"> excludes indexing by search engines and searching for links by search engines.

<STYLE TYPE="text/css"> starts the list of style attributes. HTML styles include: P, H1, H2, H3, H4, H5, H6, H7, A, and LI. P is paragraph. H1 through H7 are headings. A is anchor for name, id, and references. LI is list element. HTML styles, such as "color: blue" are in braces. text-align: center; font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 12px; font-weight: bold; font-style: italic; color: red; text-decoration: none; are examples.

<LINK HREF="stylesheet.css" REL="stylesheet" TYPE="text/css"> points to a style sheet.

</STYLE> ends the list.

HTML can use CGI to add programs. <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript" SRC="path/file.js"></SCRIPT> specifies Java script to run program. <IMG ALT="Counter" SRC="/cgi-bin/counter.pl"> specifies Counter image. <A HREF="http://.../cgi-bin/guestbook.cgi">View My Guestbook</A> uses guestbook.

</HEAD> ends heading block.

body beginning and ending

After heading block, <BODY> starts body block. <BODY STYLE="background-image: url(path/file_name.gif); background-repeat: no-repeat;"> uses the STYLE attribute to specify parameters. At line before </HTML>, </BODY> ends BODY block.

paragraph, heading, and section

<P>...</P> is for paragraph, which has blank lines before and after.

<BR> is for line feed.

<H1>...</H1> is for heading 1. Headings 2 through 7 are similar.

<DIV>...</DIV> is for text section or page division, with blank lines before and after.

<SPAN>...</SPAN> is for paragraph phrase.

<OL> <LI>...</LI> </OL> is for list.

style tag and attribute

<P STYLE="text-align: center; font-family: Verdana, Arial, Helvetica, sans-serif; font-weight: bold; font-size: 12px; font-style: italic; color: red; text-decoration: none;"> specifies parameters.

For font-family, use fonts found on all browsers, such as Courier, Times, Arial, or Helvetica. font-family:sans-serif includes Verdana, Arial, and Helvetica. font-family:serif includes Times and Courier.

For font-size, use font-size:larger, font-size:smaller, or font-size:normal.
For font-weight, use font-weight:bold or font-weight:normal.
For font-style, use font-style:italic or font-style:normal.
For color, use hexadecimal string, such as color:FFFFFF for black and color:000000 for white, or use color name, such as color:blue, color:red, or color:green.
For text-decoration, use text-decoration:none to remove underlines from links.

table

<TABLE BORDER="0" WIDTH="640" CELLPADDING="0" CELLSPACING="0" > starts table, with border, width, and cell size. WIDTH="100%" uses percent. <TR>...</TR> is for table row. <TD COLSPAN="3" WIDTH="100%" ALIGN="left" VALIGN="middle">...</TD> is for table-row detail, such as column entry or line, with column span, width, horizontal alignment, and vertical alignment. </TABLE> ends table.

image

<IMG SRC="/x/x/xxx.gif" WIDTH="15" HEIGHT="10" HSPACE="0" VSPACE="0"> is for image, with source file, width, height, horizontal spacing, and vertical spacing.

jumping

<A HREF="#Glossary">Glossary</A> jumps to same-file position marked by <A NAME="Glossary"></A>. If there is no marked position, program searches for first occurrence in file.

anchor and linking

<A HREF="path/file_name">link_name</A> is for linking to a file or URI by clicking on the link name. <A HREF="path/file_name#ID">link_name</A> is for linking to a file or URI and jumping to the ID in the file. <A HREF="path/file_name.jpg" TARGET="_blank">image</A> opens a new window. <A HREF="http://x/x/xxx.pdf" NAME="x" ONMOUSEOVER="ft(this)"></A> shows something when cursor mouses over text.

URI Escape Characters

Use % for Unicode (and ISO-Latin) hexadecimal numbers. Always encode the following URI characters. space = %20. double quotation mark = %22. # = %23. % = %25. < = %3C. > = %3E. [= %5B. backslash = %5C.] = %5D. ^ = %5E. ` = %60 = opening single quote. { = %7B. | = %7C. } = %7D. ~ = %7E. Period, apostrophe, and hyphen are OK. Do not use parentheses, underscore, exclamation mark, or asterisk.
URI reserved characters define syntax. If used in URIs, encode them. $ = %24. & = %26. + = %2B. , = %2C. / = %2F. : = %3A. ; = %3B. = = %3D. ? = %3F. @ = %40.

characters

Unicode (and ISO-Latin) characters 0 to 31 and 127 are ASCII control characters and do not print. tab = 9. carriage return = 13.
HTML recognizes the following Unicode, ASCII, and ANSI characters: blank ! double quotation mark # $ % ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; = ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [backslash] ^ _ ` a b c d e f g h i j k l m n o p q r s t u v w x y z right brace left brace | ~ ¢ £ ± µ.

HTML Escape Characters

For other characters or for escaping characters, use &#NNN; for Unicode (and ISO-Latin) decimal numbers or &...; for alphabetic. 000 to 031 have no printing characters. 009 = tab. 013 = carriage return.
032 = sp = blank = space. 033 = excl = !. 034 = quot = double quotation mark. 035 = num = #. 036 = dollar = $. 037 = percnt = %. 038 = amp = & (ampersand). 039 = apos = ' = apostrophe = closing single quote. 040 = lpar = (. 041 = rpar =). 042 = ast = *. 043 = plus = +. 044 = comma = ,. 045 = hyphen = dash = -. 046 = period = . 047 = sol = / (solidus). 048 to 057 are digits 0 to 9. 058 = colon = :. 059 = semi = ;. 060 = lt = <. 061 = equals = =. 062 = gt = >. 063 = quest = ?. 064 = commat = @.
065 to 090 are letters A to Z.
091 = lsqb = [. 092 = bsol = backslash. 093 = rsqb =]. 094 = caret = ^. 095 = lowbar = _. 096 = ` (accent grave).
097 to 122 are letters a to z.
123 = lcub = left brace = left curly bracket. 124 = verbar = |. 125 = rcub = right brace = right curly bracket. 126 = tilde = sim = ~. 127 is for Delete.
ISO-Latin characters 128 to 255 are not in the ASCII set and may not print.
Unicode does not use ANSI 128 to 159.
160 = nbsp = non-breaking space. 161 = iexcl = ¡. 162 = cent = ¢. 163 = pound = £. 164 = curren. 165 = yen = ¥. 166 = brvbar = brkbar = broken vertical bar. 167 = sect = §. 168 = uml = die = ¨. 169 = copy = copyright. 170 = ordf (superscript a). 171 = laquo = «. 172 = not = ¬. 173 = shy = soft hyphen. 174 = reg = registered. 175 = macr = hibar = ¯. 176 = deg = °. 177 = micro = µ. 178 = sup2 = superscript 2. 179 = sup3 = superscript 3. 180 = acute = ´. 181 = plusmn = ±. 182 = para = ¶. 183 = middot = ·. 184 = cedil = ¸. 185 = sup1 = superscript 1. 186 = ordm = º. 187 = raquo = ». 188 = frac14 = 1/4. 189 = frac12 = half = 1/2. 190 = frac34 = 3/4. 191 = iquest = ¿. 192 = Agrave = À. 193 =

Aacute = Á. 194 = Acirc = Â. 195 = Atilde = Ã. 196 = Auml = Ä. 197 = Aring = Å. 198 = AElig = Æ. 199 = Ccedil = Ç. 200 = Egrave = È. 201 = Eacute = É. 202 = Ecirc = Ê. 203 = Euml = Ë. 204 = Igrave = Ì. 205 = Iacute = Í. 206 = Icirc = Î. 207 = Iuml = Ï. 208 (eth). 209 = Ntilde = Ñ. 210 = Ograve = Ò. 211 = Oacute = Ó. 212 = Ocirc = Ô. 213 = Otilde = Õ. 214 = Ouml = Ö. 215 (multiply). 216 = Oslash = Ø. 217 = Ugrave = Ù. 218 = Uacute = Ú. 219 = Ucirc = Û. 220 = Uuml = Ü. 221 = Yacute = Y acute. 222 (thorn). 223 = szlig = ß (sharp s). 224 = agrave = à. 225 = aacute = á. 226 = acirc = â. 227 = atilde = ã. 228 = auml = ä. 229 = aring = å. 230 = aelig = æ. 231 = ccedil = ç. 232 = egrave = è. 233 = eacute = é. 234 = ecirc = ê. 235 = euml = ë. 236 = igrave = ì. 237 = iacute = í. 238 = icirc = î. 239 = iuml = ï. 240 (eth). 241 = ntilde = ñ. 242 = ograve = ò. 243 = oacute = ó. 244 = ocirc = ô. 245 = otilde = õ. 246 = ouml = ö. 247 (division). 248 = oslash = ø. 249 = ugrave = ù. 250 = uacute = ú. 251 = ucirc = û. 252 = uuml = ü. 253 (y acute). 254 (thorn). 255 = yuml = ÿ.

ANSI and Unicode differences

8364 (ANSI 128) (euro). 129 (not used). 8218 (ANSI 130) (low-9 quotation mark). 402 (ANSI 131) (f with hook). 8222 (ANSI 132) (double low-9 quotation mark). 8230 (ANSI 133) = ... (ellipsis). 8224 (ANSI 134) = † (dagger). 8225 (ANSI 135) = ‡ (double dagger). 710 (ANSI 136) = ˆ (circumflex accent). 8240 (ANSI 137) (per mille). 352 (ANSI 138) (S with caron). 8249 (ANSI 139) (left-pointing angle quote). 338 (ANSI 140) = Œ. 141 (not used). 381 (ANSI 142) (Z with caron). 143 (not used). 144 (not used). 8216 (ANSI 145) (left single quotation mark). 8217 (ANSI 146) (right single quotation mark). 8220 (ANSI 147) (left double quotation mark). 8221 (ANSI 148) (right double quotation mark). 8226 (ANSI 149) (bullet). 8211 (ANSI 150) (en dash). 8212 (ANSI 151) (em dash). 732 (ANSI 152) = ˜ (tilde). 8482 (ANSI 153) (trademark). 353 (ANSI 154) (s with caron). 8250 (ANSI 155) (right-pointing angle quote). 339 (ANSI 156) = œ. 157 (not used). 382 (ANSI 158) (z with caron). 376 (ANSI 159) = Ÿ.

Unicode and ANSI control characters

000 = NUL (Null character). 001 = SOH (Start of Header). 002 = STX (Start of Text). 003 = ETX (End of Text). 004 = EOT (End of Transmission). 005 = ENQ (Enquiry). 006 = ACK (Acknowledgment). 007 = BEL (Bell). 008 = BS (Backspace). 009 = HT (Horizontal Tab). 010 = LF (Line Feed). 011 = VT (Vertical Tab). 012 = FF (Form Feed). 013 = CR (Carriage Return). 014 = SO (Shift Out). 015 = SI (Shift In). 016 = DLE (Data Link Escape). 017 = DC1 (XON)(Device Control 1). 018 = DC2 (Device Control 2). 019 = DC3 (XOFF or Device Control 3). 020 = DC4 (Device Control 4). 021 = NAK (Negative Acknowledgement). 022 = SYN (Synchronous Idle). 023 = ETB (End of Transmission Block). 024 = CAN (Cancel). 025 = EM (End of Medium). 026 = SUB (Substitute). 027 = ESC (Escape). 028 = FS (File Separator). 029 = GS (Group Separator). 030 = RS (Request to Send or Record Separator). 031 = US (Unit Separator).

PDF file

Portable Document Format {PDF file} depends on a format {PostScript} (PS file) from Adobe Systems. PDF files can be unstructured, structured, or tagged, with increasing file-structure information, such as bookmarks, pages, tables, lists, and images. The freely downloadable Adobe Reader application reads PDF files. Adobe Acrobat and Adobe Distiller make PDF files.

SGML file

Standard Generalized Markup Language {SGML file} defines electronic-document structure and content. Structured documents allow searching and semantics.

XML file

Extensible Markup Language {XML file} is a Standard Generalized Markup Language (SGML).

tags

XML uses text blocks begun and ended by tags. Tags can have attributes, which can have values. Well-formed documents conform to XML language.

document type

Document Type Definition (DTD) defines document structure using tag tree. Valid documents conform to a DTD.

types

XML Linking Language (XLL) has link semantics, extended links to other documents, pointers to document parts, and links in both directions. XML Stylesheet Language (XSL) tells how to parse XML documents. Simple API for XML (SAX) allows programming, using tag-sequence events. Document Object Model (DOM) allows programming, using API-object tag trees.

languages

XML languages include Chemical Markup Language (CML), MathML, Bioinformatic Sequence MArkup Language (BSML), Biopolymer MArkup Language (BioML), GAME for Drosophila sequence information, and BlastXML for search output.

data

Modeling, storing, and querying data needs data structure. Data often grows exponentially, has complex relationships, has new data types generated from old data, has new relation types, needs archiving, has many objects with much data, has updates, and has users with different needs, skills, and tools. XML schema adds object-oriented relational database concepts. XML can be for Web interfacing among object-oriented relational databases. Only object-oriented approaches can model data. Object-oriented relational database management systems (OODBMS) have query languages, many relation types, constraints, symmetry, data clustering, many data types, tables, triggers, indexing, inheritance, security, access, methods, and objects. Relation can be not-null, unique, test, and one-of.

MATH>Computer Science>Software>Database

relational database

Databases have formatted records that allow indexing and searching. Databases {relational database} can have different-format records and relations among formats.

tables

Relational databases store data in two-dimensional tables. For example, table can be about people in a group. Rows are about one object or event, such as person. Columns are fields, properties, classes, or data categories, such as person names and weights.

tables: columns

Columns have value formats. Column values can be numbers, character strings, bitmaps, times, dates, or monetary amounts. Column values can have fixed or variable number of characters. Columns can be empty or require values. Column values can be unique or the same. Column values can be another-column values. Column values can be any sequence value, such as 1 to n.

tables: key column

Tables typically have main column {primary key}. For example, in Person table, primary key is person name or ID. Several columns can link to make primary key.

tables: index

Columns can have indexes. Indexes assign table-row number to column value. Columns can require that each value occur only once or allow values to repeat. For example, person names occur only once, but the same weight can apply to several people. It takes time to make indexes, but indexing speeds searches. Indexing is more efficient if table updates are infrequent.

tables: relations

Two tables cross-reference each other, if they both have same column. For example, Person table can have PersonID column and WeightCategory column. Weight table can have WeightCategory column and WeightAmount category. Both tables share WeightCategory column. Database queries that use both tables can yield WeightAmount for PersonID. WeightCategory column in weight table is reference {foreign key} for WeightCategory column in Person table. Database can allow only weight categories listed in weight table to be values for WeightCategory column in Person table.

trigger

Inserting, deleting, or updating table values can cause {trigger, database} actions on database tables. For example, if Person table lists all people and their groups and if group table lists group members, adding person and his or her group to Person table can automatically add person to Group table.

normal forms

Theory governs table structure in relational databases. Five principles {normal form, database} improve performance speeds and prevent update problems, data redundancies, and data inconsistencies. Tables and records typically have normal forms {normalization, database}, unless performance considerations require non-normalization.

normal forms: First Normal Form

Table records have same number of non-repeating fields. For example, people in Person table have names, weights, and heights, even if some people have no known height values.

normal forms: Second Normal Form

Tables have one primary key. In Person table, name field is primary key, because it identifies person uniquely. Other fields state facts about primary key, such as weight and height, and are not unique.

normal forms: Third Normal Form

Columns are independent of other columns. There is only one column about each property. For example, in Person table, there are not two columns about weight, one for kilograms and one for pounds, or one for weight numbers and one for weight categories. For weights, there can be a Weight table. Non-key fields cannot be subsets of, or have another relation to, primary keys. For example, Person table has name column and does not have nickname column. For nicknames, there can be a nickname table.

normal forms: Fourth Normal Form

Row and column cells have one value, not multiple values. In Person table, name column has only best name, not alternative names, and language column has only preferred language, not all languages spoken. For nicknames, there can be a nickname table. For languages, there can be a language table. In Person table, each column is only about one property, not multiple properties. For example, Person table does not have a name-language column. For names and languages, there can be a language table.

normal forms: Fifth Normal Form

If two columns relate, and relation does not have direction {symmetric constraint}, use several tables with as few fields as possible, rather than fewer tables with more fields. For example, to relate person, weight amount, and weight category, Person table has name and weight category columns, and weight table has weight amount and weight category. Person table does not have person, weight amount, and weight category columns. For asymmetric constraints, database must have two tables.

tablespace

Single files {tablespace} can store related tables. Tablespaces represent shared columns only once, so memory size and computer processes decrease and processing speed increases.

Tablespace tables {cluster} {data cluster} can share columns. Clusters place related information in one physical location and allow overlapping indexes. Clustering works well for adding data to tables but is not good for tables that have many deletions or updates.

pivoting of data

For tables, cell values in any column or row can become columns or rows {pivoting}|. The new table shows a different perspective on data, along a new dimension. Pivoting can correlate two table variables.

empty cells

Cells can be empty (NULL). Pivoting treats NULL the same as other cell values. Pivoting can make fewer cells and so fewer NULL cells.

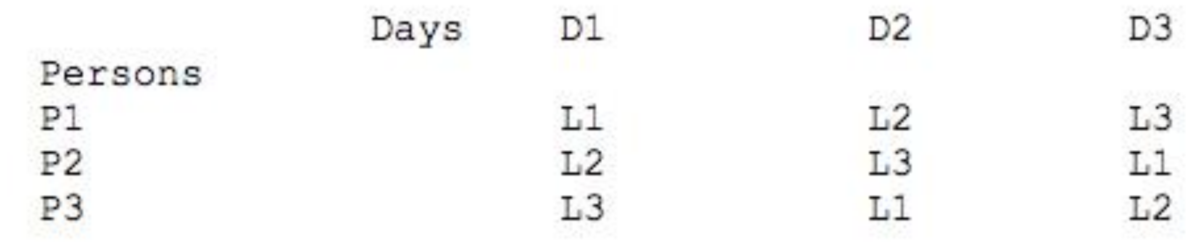
example

Table has Days columns D1, D2, ... and Persons rows P1, P2, ... Table cells have Locations L1, L2, ..., with L1 in D1-P1 cell, and so on. See Figure 1. You can pivot table to make table with Days columns, Locations rows, and Persons in cells. See Figure 2. Cells fill by rule: Days and Locations have Persons. You can always pivot new table back to old table.

data rows

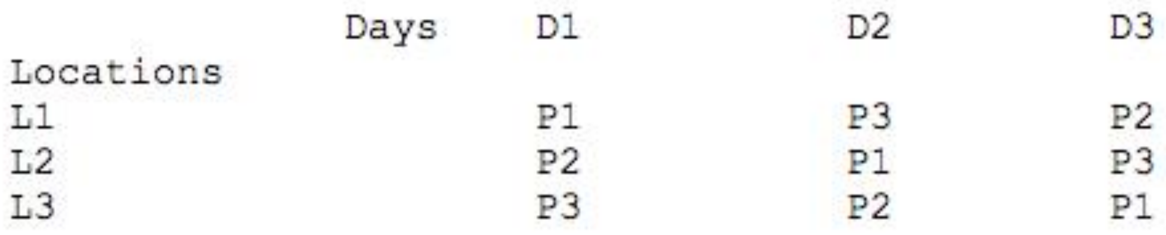
You can pivot table on one column to put another column in cells. For example, table can have Day, Location, People, Thing, and Amount columns. Row records are Days, Places, People, Things, and Amounts combinations. You can pivot on Day column and rows, to put Amount in cells. Now table rows are Thing, People, and Place combinations, and columns are Day values listed in original-table cells. Cell Amount matches Day, Place, People, and Thing. If new table has same or greater cell number as original table, you can pivot back to original table.

Figure 1



	Days	D1	D2	D3
Persons				
P1		L1	L2	L3
P2		L2	L3	L1
P3		L3	L1	L2

Figure 2



Days	D1	D2	D3
Locations			
L1	P1	P3	P2
L2	P2	P1	P3
L3	P3	P2	P1

security policy

Databases need security {security policy}: database roles (USER and ADMINISTRATOR), role system privileges for each application, role object privileges for each application, users, and row-level security.

Application Security creates roles and grants them privileges for one application. Roles and privileges for other applications should not affect the application. Users should not be able to alter roles and privileges inside or outside of the application. Application security must include SQL*Plus, SQL*Loader, and all other such available applications which might allow unrestricted SQL statement execution.

Application Context sets up session-based attributes. Each user has attributes, such as a user name, a role, an application, tables and views authorized to access, and other information which can be used to determine context. The context is used to determine what is accessed in the session.

Access Control allows row-based security for tables and views using Oracle functions attached to the table or view which is the basis of an application. When a user enters a SELECT, INSERT, UPDATE, or DELETE statement on the table or view, Oracle modifies the statement to include a WHERE clause which limits the data affected.

A schema is owned by a user. Schemas are accessible to other users when privileges are granted by the owner. The USER is the schema which contains all objects.

triggering

Database change or command use {event, computer} can trigger action {triggering, database}, such as displaying report or executing program.

MATH>Computer Science>Software>Database>Data Versioning

version number

Every time data changes, it can have higher identifying number {version number}. Databases can keep previous record versions {version control}.

generation in versioning

Version branches {generation, data} start with generation 1. Generation and version combinations are unique.

MATH>Computer Science>Software>Command

command to computer

Commands {command, computer} {executable command} execute actions, such as login, logout, show user name, or change password.

execute command

Commands {execute command} can save data to database, like Commit, and start background processing.

commit command

Commands {commit} can save data to database. Commit saves everything since previous Commit command, Rollback command, or application opening.

rollback command

Commands {rollback} can discard all unsaved data and return to state at previous Commit command or application opening.

login

Commands can allow user into computer system {login}.

logout

Commands can allow user to exit computer system {logout}.

MATH>Computer Science>Software>Component

component of system

Software parts {component, computer} can call other applications, products, or components.

Application Programming Interface
Components {Application Programming Interface} (API) can share inputs and outputs.

Database Management System
Components {Database Management System} (DBMS) can store and retrieve database data.

Enterprise Java Beans
Component architecture {Enterprise Java Beans} (EJB) can deploy object-oriented distributed enterprise-level applications. Applications using Enterprise Java Beans architecture are scalable, use transactions, and can have multiple users.

Graphical User Interface
Users see window sets {Graphical User Interface} {user interface} (GUI) (UI), containing text fields, selection lists, check boxes, buttons, and other controls, in which to enter or view data and to issue commands.

module in software
Software components, applications, and products have subsets {module, software} {software module} that perform commands, actions, functions, or operations.

MATH>Computer Science>Software>Security

computer security
Computation can have constraints on users and data {security, computer} {computer security}.

A secure computing environment involves identifying a user at logon or when accessing services (authentication) and assigning privileges to directories and files to users or groups of users (authorization).

A security policy establishes administrative procedures for reacting to break-ins, backing up data, logging on, allowing user access, and setting appropriate security levels for hardware, software, application programs, and network services.

A security configuration for a computer involves the hardware, software, application programs, and network services and includes administrative procedures, physical security, installation, network access considerations, user training, and software security, such as user accounts, object permissions, auditing, and registry keys.

C2 Security Level
The National Computer Security Center (NCSC) of the United States performs software security evaluations for operating systems and publishes the results in a book entitled "Department of Defense Trusted Computer System Evaluation Criteria, " also known as the "Orange Book."

The C2 security level has the following main requirements:

The owner of a resource (such as a file or directory) must be able to control access to the resource.

The operating system must protect objects from unauthorized reuse by other processes. (As examples, freed memory contents cannot be read by another process, and data from deleted files cannot be accessed.)

To access the system, each user must enter a unique name and password. The system must be able to track all user activities using the unique identification.

System administrators must be able to audit security-related events, and access to audit data must be limited to authorized administrators.

The operating system must protect itself from external interference or tampering (As examples, modification of a running system or modification of disk files is prevented.)

The NCSC document "Introduction to Certification and Accreditation" discusses how to configure the physical and software computing environment. Certification is the use of the computer system in a specific environment. Accreditation is an evaluation of the certification plan by administrators.

Network Security
Network security evaluations are contained in the "Red Book" of the NCSC.

Network
Isolate the workstation from the network if possible. Preferably use a network entirely contained in a secure building. If network cables are in unsecured areas, use optical fiber cables to prevent wiretaps.

Internet

Do not connect the workstation to the Internet.

FTP

If the workstation is on the Internet, file transfer protocol (FTP) might be available. FTP allows public file access via anonymous log on. There is a user account for anonymous logons (GUEST by default) and a default home directory.

If FTP must be made available, create a user account specifically for FTP and require a password. Do not assign the account to any user groups. Do not allow the account the user right to "Logon on Locally."

As the home directory, assign a separate disk partition, with no secure files or directories.

Remove the NetBios Interface.

Security Administration

System administrators should follow specific rules for effective computer security:

Allow few users to have Administrator privileges. Always log out of the Administrator account when not using the workstation.

Set computer security to the minimum level required to protect resources to minimize setup, maintenance, and security evasions (such as password sharing).

Document backup and all other security procedures to create a security policy.

Always follow the procedures of the security policy.

Monitor security logs regularly and adjust security policies and settings as needed.

Use an up-to-date anti-virus and anti-Trojan Horse program.

Use security mailing lists to be aware of virus attacks.

Check software vendors for recent security fixes.

Always update software with the latest service pack.

Use the Backup utility to back up the registry, files, and directories regularly. Allow Backup privileges only to Administrators.

Set the security features of applications to work with computer security features.

Security administration for a workstation should include physical security, installation, network security, user training, and software security (such as user accounts, object permissions, auditing, and registry keys).

Physical Security

Physical security involves computer and cable location and locking, physical access to the CPU and removable media, power on and rebooting from floppy disk or network, testing of the CPU and peripherals, hard disk set up and maintenance, and surge protection.

Access

Protect the computer workstation from theft. Place the computer in a case that requires a key to open and store the key away from the computer. Attach the computer to a wall or desk with a locked cable. Allow access to the room only to authorized persons. Establish procedures for moving or repairing the computer.

Expose only the computer's keyboard, monitor, mouse, and printer to users. Prevent unauthorized users from touching the power or reset switches, the CPU, and the removable media drives by locking the case or locking the entire CPU in a ventilated enclosure. Access to drives can be limited by using a locked opening.

Booting

Use a workstation with a power-on password to prevent unauthorized personnel from starting another operating system. Set the power-on password using the vendor's instructions, typically by changing the computer's BIOS setup.

Remove the floppy disk drive or disable a floppy-based boot. Also, remove the network card if possible. Allow booting only from the fixed disk.

Edit the boot.ini file so that the boot time-out is 0 seconds, to prevent booting another operating system.

Lock the CPU to physically prevent another operating system from being loaded.

Testing

Test the processor using diagnostic integrity tests.

Test peripherals using the integrity tests of the product.

Hard Disk Maintenance

Make the entire hard disk one file system. Do not use the FAT file system, which has no security features.

Scan and defragment disks regularly to isolate bad sectors and maintain performance.

Surge Protection

Use a surge protector or power conditioner to protect the computer and its peripherals from power spikes.

Installation

Use a password at least nine characters long with some non-alphabetic characters in the first seven characters.

Create an Emergency Repair Disk (ERD) and lock the disk in a secure area (the ERD contains security information).

Set up a special location called the Alternate Data Location, recorded in the Registry.

For Network Protocol, select only TCP/IP.
Use a fixed IP address. Do not use DHCP.
Add the workstation only to a secure domain.
Do not install by disk duplication (which uses the xcopy command).
Do not install any other operating systems on the computer.
After installation, reboot Windows and log on as an Administrator.
Check the Installation Report.

User Training

User training involves use of passwords, proper log on, file protection for copies, data deletion at log off, necessary restarts, proper log off, and use of XXXX file checksums.

Passwords

Users should follow the password rules:
Memorize passwords (never write them down).
Use passwords longer than five characters.
Change passwords frequently.
Never reuse previous passwords.
Never use easily-guessed passwords or words in a dictionary.
Use a combination of letters and numbers.

Log On

Users should always press CTRL+ALT+DEL to log onto Windows, in order to obtain the secure Windows log-on screen. Other logon methods or dialogs risk use of a substitute log-on screen designed to collect account passwords.

File Copies

When a file is copied to a new directory, the file acquires the permissions of files in the new directory. The protections assigned to the document in the old directory are lost.

After copying a file to a new directory, users should set the protections on the file, or alternatively, first move the file to the new directory and then copy the file back to the original directory to reset the protections.

Data Deletion

Users should remove sensitive data from floppy disks or CD-ROM drives before logging off.

Restart

Users should restart the computer before using the tape drive.

Log Off

Users should log off or lock the workstation when not at the computer. Users should set the workstation to lock automatically if not used for a period of time. A 32-bit screen saver with the Password Protected option has this feature.

Checksums

All data files include a checksum value. Compare file checksums to determine if the file has been modified by any other application or any outside process.

Software Security

Software security involves user and group accounts, auditing, file and directory permissions, and registry key settings. Software security protects against accidental or deliberate changes to the data files, the operating system software, and the computer set up and security configuration.

Check the user accounts, security settings, trust relationships, network access points, and the registry regularly. Check for inconsistent permissions of users and objects, user accounts that should be disabled, and unauthorized network connections.

User and Group Accounts

Create users and user groups. Assign users to user groups.

Password and Account Lockout Policies

Set password and account lockout policies for user accounts.

Set the maximum password age to 42 days. Allow the password to change immediately. Set the minimum password length to six (or more). (Make sure that the Permit Blank Password box is not checked.) Set the password uniqueness so that no password history is kept.

Set account lockout to allow only three invalid logon attempts, to reset the count after 60 minutes, and to have a duration of 60 minutes.

User Rights and Auditing Policies

Give each user or user group user rights.
Grant privileges to objects and procedures.

File and Directory Permissions

Control access to files and directories.

Event Auditing

Select auditing options and Audit Policies.

Alerts

Set administrative alerts (electronic mail messages) to System Administrator if any attempt at unauthorized use or other security problem occurs.

digital rights management

Software {digital rights management} (DRM) can prevent copying files.

rootkit

Applications {rootkit} at root-user level can hide program files, such as copy-protection programs.

row-level security

Methods {row-level security} can control access to project data based on role assigned to a user by the administrator, because projects and roles are linked. The table privileges granted to a role are removed. A package has a function to make the predicate for the queries. Policies use the predicate-creating function on all statements (SELECT, INSERT, UPDATE, DELETE) on the project tables. Depending on the role assigned to the user and the projects paired with that role, users logging in will only see the protocols or experiments associated with the allowed projects.

MATH>Computer Science>Software>Security>Challenge

computer challenge

Applications can request password, fingerprint, or electronic signature {challenge, computer} {computer challenge}, before allowing access to information or functions.

electronic signature

Users must enter username and/or password {electronic signature} before committing data to database. A field holds a unique code based on user ID and password. Electronic signatures include the user name, date, time, role of signer (review, approval, authorship, or ownership), and digital signature. Digital signatures are generated using biometrics or cryptography.

Saving files is the typical example of an event that requires an electronic signature. Creating an electronic signature displays the role to be applied and requires entry of the user name and password; if multiple signatures are required in one user session, later signatures require only a password. Electronic signatures are only assigned to verified individuals. Electronic signatures cannot be assigned to someone else. Persons with electronic signatures certify that their electronic signatures are to be legally equivalent to handwritten signatures.

MATH>Computer Science>Software>Security>Data

auditing of data

Programs {auditing} can record who changed database data, when, and why, as shown in audit logs.

keychain for data

Users have identification and authentication information, plus information about what user can access {keychain}.

profile of computer user

Users can have permission to execute specific functions {profile, computer}.

static data

Users do not update system data {static data}, such as instrument locations, entered by administrator at system setup or upgrade.

MATH>Computer Science>Software>Security>Users

user of computer

People {user, computer} can access software or data, after authentication.

root user

Users {root user} can control whole computer.

computer administrator

Users {administrator, computer} {computer administrator} can have access to whole system have privileges to add, remove, or edit all data and manage access control for other users.

datagroup

Users can access only data in their datagroups {datagroup}.

role for user

Users have permission to execute specific functions {role, computer}. Roles control user access to application functions. Interfaces can display available commands and not display unavailable commands.

security group

User groups {security group} can access data types in special ways: read/view, update, create, and/or delete.

user group of computer

Users, such as technicians, administrators, scientists, or guests {user group}, can have same privileges to access data and functions.

MATH>Computer Science>Software>Operating System

operating system

Software {operating system}| can allocate CPU resources. Operating systems determine peripheral number and type. They handle input-output operations, manipulate and process files, manage and allocate memory, begin and end jobs, and keep accounting, data-loading, job-priority, error-checking, backup, and recovery records. They put system description {system configuration} in memory. They can execute several programs at once {multiprogramming}, using swapping and executive control. They can allot maximum time {timeslicing} to program before switching to another. Operating systems can have application programs, which perform business or fundamental functions.

Unix

The most common operating system {Unix} has many variations. At login prompt, enter username. At password prompt, enter password. To change password: passwd followed by old password, and repeat new password. To log out: logout. To create a directory: mkdir directory-name. To display file: cat filename or more filename. To print file: lpr filename. To list files and directories: ls directory-name or filename. To rename files and directories: mv source-filename destination-filename, or mv source-filename destination-directory, or mv source-directory-name destination-directory-name. To copy files: cp source-filename destination-filename or cp source-filename destination-directory. To delete file or directory: rm filename or rmdir directory-name. To change working directory: cd directory-name.

Pathnames use forward slash. ~ is Home directory. . is working directory. ..is parent of working directory.

wild cards

? is single character wild card. * is arbitrary number of characters.

redirecting

command-name > filename - System types output of command to file rather than screen, replacing current contents of file, if any.

command-name > > filename - System types output of command to file rather than screen, appending to current contents of file, if any.

editing files

vi - enter text editor

search files

grep search-string filename to type lines containing the string in a specific file.

online documentation

To see online Man Pages: man command-name.

MATH>Computer Science>Software>Operating System>Language

assembler
Operating-system programs {assembler}| can change programs into assembly-code language.

compiler
Operating-system programs {compiler}| can change high-level language program into computer language used by computer hardware, such as binary code.

interpreter program
Operating-system programs {interpreter program} can translate instructions from high-level program to binary code and runs computer program immediately.

MATH>Computer Science>Software>Operating System>Parts

controller module
Operating-system programs {controller module} can control operations.

data management program
Operating-system programs {data management program} can manage data flow.

diagnostic program
Operating-system programs {diagnostic program} can diagnose problems.

emulator
Operating-system programs {emulator} can represent another computer.

executive control program
Operating-system programs {executive control program} can control several programs and interfaces.

format maker and user
Operating-system programs {format maker and user} can print or display formatted job output.

generator program
Operating-system programs {generator program} {object program} can make object programs from parameters.

initial program loader
Operating-system programs {initial program loader} can start computer system by placing operating system in memory and starting CPU.

job scheduler
Operating-system programs {job scheduler} {job control language} can perform program commands most efficiently.

maintenance program
Operating-system programs {maintenance program} can be for on-line testing.

memory dump program
Operating-system programs {memory dump program} can read memory contents.

operator routine
Operating-system programs {operator routine} can load programs.

simulator
Operating-system programs {simulator} can represent models.

sorting device
Operating-system programs {sorting device} can organize different memories.

utility program
Operating-system programs {utility program} can track or organize computer system and memory.

workflow language compiler
Operating-system programs {workflow language compiler} can accept and store jobs.

MATH>Computer Science>Software>Operating System>Operations

computer operations
Computers have operations {computer operations}. Programs can run immediately {interactive processing} {on-line processing} {real-time processing}. Programs can execute later, typically at night when CPU usage is low {batch processing, computer} {off-line processing}. On-line transactions take time {response time} {flowtime} {float time} to process [Luce, 1986].

multiprocessing
Computer systems can have two or more processors {multiprocessing}. Job-control program decide job types to do and their priority and puts jobs in job queue.

swapping
Operating systems can add and remove program parts {swapping}| {memory swapping} from storage devices into memory {virtual memory}, to use only needed program sections. Virtual memory uses either fixed-length pages or logical program segments.

throughput
Operations process many characters per second {throughput}|.

timesharing
Several users can use CPU at once {timesharing}|, because CPU is fast compared to input rates.

MATH>Computer Science>Software>Algorithm

algorithm
Ordered instruction sequences {algorithm}| {recursive procedure} can perform functions in finite numbers of steps. Algorithms start from input data, process data, and make output data. Algorithms use finite numbers of symbols. Computing is algorithm automation and execution.
components
Algorithms use non-algorithmic components, such as containers, container adapters, function objects, and file manipulations. Containers are vectors and matrices. Container adapters are point graphs and edge graphs. Function objects are for sorting, generating random numbers, having properties, and comparing numbers.
input
Input requires redundancy to overcome noise.
output
Performing algorithm gives results {enaction}. Results are in code and have no intrinsic meaning. Result meaning relates to coded input.
processing
Algorithms can be recursive and nested and use feedback and feedforward. The shortest way to predict algorithm results is to run algorithm. The most compressed algorithm description is algorithm itself.

Church thesis for functions
Computable functions can be partial recursive functions {Church's thesis, algorithm} {Church thesis, algorithm} {Turing's thesis}.

MATH>Computer Science>Software>Algorithm>Kinds

approximation by discrete

Using time-interval midpoints, discrete functions can approximate continuous or non-discrete functions {approximation by discrete function} {discrete function approximation}.

non-deterministic computer

Ideal computers {non-deterministic computer} can choose better at decision points. Real computers {deterministic computer} choose by approximate methods.

non-linear processing

Processing {non-linear processing} can describe quantity interactions. Multiplications combine two or more different measurable things to get new thing. Multiplication can reduce data from several quantities to one quantity. For example, gravitational force multiplies two interacting particle masses and then divides by distance between them. Multiplicative effects include division, differentiation, and integration. Differentiation reduces interaction number. Integration increases interaction number.

MATH>Computer Science>Software>Algorithm>Kinds>Analysis

forward analysis

Analysis algorithms {forward analysis} can use assumptions about node relations to go from one node to another, to avoid costly and time-consuming backtracking.

frame analysis

Analysis algorithms {frame analysis} can use object-class variable, parameter, or property-type lists, with default values. However, such lists do not use "OR" relations well.

MATH>Computer Science>Software>Algorithm>Kinds>Search

alpha-beta technique

For hierarchies or networks, heuristic search methods {alpha-beta technique} {minimax algorithm} repeatedly evaluate nearby situations to check if they are nearer goal situation.

scoring

Alpha-beta technique assigns scores to all evaluated situations {node, hierarchy}. Score determines which nodes to check further.

node

Nodes have immediate-successor nodes. Alpha-beta technique checks beyond current nodes minimum distance and maximum distance {depth, search}. Node scores decide which immediate successors to check further. Alpha-beta technique removes low-scoring nodes {pruning} {alpha-beta pruning} or progressively worse nodes {tapered pruning}.

depth

Alpha-beta is depth first search, rather than width first. Alpha is maximum lower bound and can only increase. Beta is minimum upper bound and can only decrease.

generate and test algorithm

Techniques {generate and test algorithm} for reaching goal situations can produce random or directed trials and check results.

Grover algorithm

Subtract searched-for message wavelet from possible wavelets to find wavelet that makes zero {Grover's algorithm} {Grover algorithm}. Invert wavelet. Probabilities stay the same. Find average amplitude. Invert wavelets around average amplitude. Searched-for wavelet magnifies amplitude while others reduce. Repeating converges on searched-for wavelet.

heuristic search

For hierarchical networks, searches {search, computer} {computer search} can go first to lower nodes {depth first search}, stay at same level {breadth first search}, or use evaluation function {heuristic search}|. Heuristic search tries to reach goal. It starts from current situation, evaluates nearby situations to short depth to see if goal is closer or farther, chooses only nearby situations that are closer, and then repeats, until exhausting all new nearby situations or finding goal.

evaluation

Evaluation uses scoring methods. Searching requires assigning probabilities. Search eliminates nodes by checking if they have successors, by assigning scores, or by checking for duplicates. Search can minimize risk {minimax, search}. Heuristic search can evaluate search steps to decide whether to go backward or forward {Shannon paradigm}. Search can use other strategies {set-of-support strategy} {unit preference strategy} {linear input form strategy} {ancestry-filtered strategy}.

hill climbing

Techniques {hill climbing}| for reaching goal situation can find path from initial to goal situation, while always maximizing score. Hill climbing locally improves answer until reaching optimum. Hill climbing often stops at solution that is not the best globally, because possible solutions are independent and are in different space locations, with valleys between them. Better hill climbing requires overview to check larger regions.

hypothesize and match algorithm

Techniques {hypothesize and match}| for reaching goal situation can make many examples to compare to true goal {match, search} or make and iteratively test better hypotheses until one matches.

resolution procedure

For reaching goal situations, techniques {resolution procedure}| {resolution principle} can check if all formulas are true and check if negative of formula to prove is unsatisfiable, to imply formula. Resolution procedures work if formulas, expressed in predicate calculus, describe situation space. Formulas must allow causality, ability, temporality, recursion formulas, and spatial relations.

situation space

Problems can have initial situations, possible situations, actions moving from one situation to another, action relations, and goal situations {situation space}|. Solving problem involves finding path from initial situation to goal situation, while avoiding traps.

system inference model

Techniques {system inference model} for reaching goal situations can find function or relation between initial situation and goal situation or build relations using examples and counterexamples.

MATH>Computer Science>Software>Algorithm>Polynomial

polynomial time

Problems {P problem} can be solvable with polynomials {polynomial time}| {P time}. Time to solve them increases as polynomial-first-term power, as number of items grows. Linear growth has power one: n^1. Quadratic growth has power two: n^2. Cubic growth has power three: n^3.

Polynomial-time problems include whether number is prime or not, if two lists have all members paired, and if all towns on maps have paths to other towns.

exponential

Problems {non-deterministic problem} {NP problem} can grow superpolynomially or exponentially in time, resources, or memory {non-deterministic polynomial time} {NP time}, as number of items grows. NP problems include all P problems.

Konigsburg bridges problem is an NP problem. Superpolynomial problems {subset sum problem} can try to find sets of numbers that add to zero from a large number set. Can a set of boxes fit in a larger box? Do three-color maps have no neighboring countries with same color?

Evaluating proposed solutions to NP problems requires polynomial time.

NP-complete

NP problems {non-deterministic polynomial time complete} {NP-complete} can be the most general. NP-complete problems include all NP problems. Quantum computers cannot solve NP-complete problems in P time, but can solve some NP problems in much faster NP time, by using specific problem features.

satisfiability problem

People have conflicting demands, so a problem is how to satisfy people {satisfiability}| {satisfiability problem} (SAT). Satisfaction depends on others' choices. Satisfiability problems are NP-complete, because they scale exponentially.

MATH>Computer Science>Software>Finite-State Machine

finite-state machine

Software machines {finite-state machine} can have a finite number of possible states.

automata theory

Finite strings can map to finite strings {automata theory}.

symbols

Systems have finite symbol sets, such as 0 or no symbol and 1.

strings

Finite tape has segments, each with one symbol. Segment series make symbol series.

read and write

Tape reader reads one tape segment. Tape writer writes one tape segment. Tape mover moves tape one segment forward or backward.

tape

Tape has input symbols and starts at initial position.

state

Automaton starts in initial internal state. Automaton changes state in response to rules. Controller has a finite number of possible states.

rules

Given current state and symbol, functions {transition function} {next-output function} tell which symbol to write and where, functions {next-state function} tell what next controller state is, and functions {next-move function} tell whether to move tape, read tape, or write on tape.

processing

Controller decides what tape mover, tape writer, and tape reader do at next step, based on symbol reading, controller state, and rules. When automaton stops, original string has become new string.

algorithm

Automatons can represent any computer program or algorithm.

MATH>Computer Science>Software>Graphics

computer graphics

Image files {vector file} {geometric file} can use mathematical formulas that software {object-oriented graphics} {vector graphics} {computer graphics} {graphics} {draw program} manipulates. Vector-graphics images transform easily, look good on higher-resolution displays, and use less memory than bitmaps.

fonts

Scalable fonts {True Type font} {vector font} {scalable font} {object-oriented font} {outline font} use vector graphics.

parameters

Main parameter is geometric-object type, such as rectangle. x, y, and z coordinates locate figure. Width and height are dimensions. Lines have color and size. Fill has color, opacity, orientation, and pattern.

spline

Splines represent lines and can represent boundary lines that divide surfaces into polygons.

surfaces

Closed surfaces can be polygons, such as triangles. Surfaces can be parallel lines or grids.

volumes

Splines and generalized ellipses or ellipsoids can represent volumes.

formulas

Vector graphics uses formulas for points, lines, splines, polygons, circles, ellipses, Bézier curves, Bézier splines {bezigon}, text in Bézier curve fonts, Catmull-Rom splines, NURBS, and rectangle bitmaps. For example, circle information includes radius, center point, line style, line color, fill style, and fill color. Formulas can combine.

formulas: transformations
Most formulas allow transformations, such as rotation, translation, reflection, scaling, stretching, and skewing. Programs can perform logical and set operations. Programs can also add surface-texture bitmaps, add shading, and smooth irregularities.
bit map
Images can be pixel arrays {bit map} {raster file} that software {raster graphics} manipulates. Programs {paint program} can make bit maps smaller or larger, but enlarged images have blank or repeated areas. Vector graphics convert to raster graphics for dot-matrix printers, laser printers, and display monitors, but not plotters.

Bezier curve
Computer-graphics algorithms {de Casteljau's algorithm} {Farin-Boehm algorithm} can evaluate parametric curves {Bézier curve} or surfaces {Bézier surface}. Linear, quadratic, and cubic Bézier-curve {Bézier polygon} splines can draw any shape.
history
Paul de Casteljau invented de Casteljau's algorithm [1959]. Pierre Bézier of France designed automobiles [1962].
equations
Lines can use two points P0 and P1 {control points} and have equation f(t) = (1 - t) * P0 + t * P1, where 0 <= t <= 1. Quadratics can start at P0, go toward P1, and end at P3: f(t) = (1 - t)^2 * P0 + 2 * t * (1 - t) * P1 + t^2 * P2, where 0 <= t <= 1. Cubics can start at P0, go toward first P1 then P2, and end at P3. f(t) = (1 - t)^3 * P0 + 3 * t * (1 - t)^2 * P1 + 3 * t^2 * (1 - t) * P2 + t^3 * P3, where 0 <= t <= 1.
interface
Drawing programs display control points, which control lines.

level set method
Real functions can have domain-value sets {level set} {fiber, set} that make function constant. Real-equation roots are level sets. For two-dimensional domains f(x,y), level sets are curves {implicit curve} {isocontour}. For three-dimensional domains f(x,y,z), level sets are surfaces {implicit surface} {isosurface}.
function
Functions are positive inside surface, zero on surface, and negative outside surface. At points, function gradient is perpendicular to level-set surface.
shape
Closed planar curves can be isocontours. Closed surfaces can be isosurfaces. Shape boundaries are level sets. Shapes have real-function positive values. As object changes, level-set function translates by non-parametric methods {level set method} {Eulerian approach} (Stanley Osher and James Sethian) [1988].

ray tracing
Computers can test all rays entering screen to see where they land on object-indexed kd-tree scene {ray tracing} {ray processing unit} (RPU). Ray tracing can check for shadows, reflection, refraction, and color.

graphical processing unit
Rasterization software renders three-dimensional scenes for two-dimensional screens {graphical processing unit} (GPU). GPU first makes scene into independent plane polygons, typically triangles. It must find which ones are closer {z-buffer}. Because light reflects from objects onto other objects {global illumination}, GPU must later add indirect light.

MATH>Computer Science>Software>Programming

computer programming
Steps in making computer programs {computer programming} {programming, computer} are defining problem, creating algorithm to solve problem, making flowchart to see how to implement algorithm conceptually, writing program using language and following flowchart, and checking program for errors.

computer program
Computers can process problem-solving algorithms {computer program}. Computer programs are explicit formal processes to generate output from input and internal settings. Programs have instruction sequences.
types

User programs perform arithmetic calculations, word processing, or spreadsheet manipulations. Utility programs perform functions about files and directories. Environmental programs work in background to perform accounting, memory-utilization, and scheduling functions.

making

Set goals or requirements. Develop functions to achieve goals. Design how to implement functions. Implement actual code.

module

Programs can have small independent programs {module, program}, to reach intermediate goals.

errors

Languages have special commands or techniques for error or exception handling, which replace IF ... THEN ... statements, prevent crashes, give needed information about errors or warnings, and provide logging information.

process

Code development follows standard process, uses standard methodology, and has life cycle. Requirements state involved units and required deliverables. Flow charts, such UML diagrams, show workflows, inputs, outputs, and use cases.

program structure

Programs have subunits {program structure}, such as declarations, executable units, loops, functions, subroutines, procedures, and recursive or reentrant subunits. Program have data types, such as arithmetic, Boolean, alphanumeric, formal or algebraic, string, list, vector, array, or hierarchical. Program statements assign variables or values, handle alphanumeric data, control sequences, make decisions, input, output, debug, store, and operate on system statements.

addressing in program

Programs can use addresses {descriptor, computing} to locate program segments or data blocks {addressing}. Descriptors contain addresses pointing to program-segment or data-block locations, word counts indicating program-segment or data-block sizes, and descriptor-sequence descriptor bits {presence-bit}.

collection in program

Sets {collection} can have same-type objects {homogeneous collection} or different types {heterogeneous collection}.

debugging in program

Checking {debugging}| programs for errors follows programming.

decision table

Stored value arrays {decision table} {lookup memory} can allow result look-up for already-performed calculations, rather than performing calculations.

instruction in program

Processing operations {instruction, computing} need two steps. First, get instruction from storage and put it in instruction register. Then get data from address and put it in data register. Instructions can specify many addresses, including next instruction address. CPU has 30 basic instructions.

stack

Instead of using addresses, instructions {zero-address instruction} can specify stack, stack dictionary level, and dictionary-level index value {address couple}.

program

Instructions are elementary switches or gates {microfunction} and so are small programs {microprogram}.

open-closed principle

Existing functions can be open for extension to new requirements but closed for modifications affecting published interfaces {open-closed principle}.

operations in program

Operations {operations, program} on input values give output values. Operations include all arithmetical operations. Operations {let command} {set command} can calculate function value. Operations {if ... then ... commands} {conditional command} can check statement truth and perform actions. Operations {for ... next commands} {loop command} can repeat instruction sets over index-value ranges. Operations {goto command} {subroutine} can jump to another instruction block.

Polish notation

Statements can have operators and operands in correct sequence to execute them {Polish notation}, requiring no addressing, storage, or retrieval.

set-based analysis

Program variables can be infinite value sets {set-based analysis} (SBA), so program ignores variable dependencies but considers all data and data structures.

string operation

Alphabetical operations {string operation} can extract sequence {string, text} parts {substring}. String operations can find string location {index, string} in another string. String operations can partition string to make array.

workflow of system

Logical-step series {workflow}| start from input and use processes and decisions to reach output, typically using branching paths.

MATH>Computer Science>Software>Programming>Architecture

software architecture

High-level function-organization description {software architecture} {system architecture} defines system.

computer system

Components include applications, technical services, integration services, business services, and data services {computer system}.

Common Object Request Broker Architecture

Architectures {Common Object Request Broker Architecture} can have object-oriented ways to handle data between clients and servers, using Interface Definition Language (IDL) and Internet Inter-ORB Protocol (CORBA).

envelope in software

execution environment {envelope, computing}.

implementation model

Models {implementation model} can describe function logical and physical representation.

overlaying in software

Modules can have use in several programs {overlaying}.

punch-out in software

User-interface sections {punch-out} can be for multiple applications.

Remote Method Invocations

Method invocations {Remote Method Invocations} (RMI) can use Java data model.

toolkit in software

Related reusable class sets {toolkit} provide general functions.

unified process software

Integrated project stages, methods, and principles {unified process} develop and maintain software.

value object

Objects {value object} transfer data between presentation and domain layers.

MATH>Computer Science>Software>Programming>Architecture>Design

software design tools

Tools {software design tools} can check high-level design models {model checking}, using automated reasoning on structure language, to prove algorithms have no counterexamples or contradictions. Integrated-circuit design uses model checking. Systems have object operations, mappings, and relations, with assertions. Design-checkers try to find cases in which assertions are not true, so designers can exclude those cases.

contract-based design

Designs {contract-based design} can specify contracts between user and service provider. If user provides requisites {pre-conditions}, then service provider guarantees service will perform functions and give results {post-conditions}.

responsibility-driven design

Starting with requirements, specifying functions and designing objects {responsibility-driven design} can meet requirements.

top-down design

Designs {top-down design} can use modules, which contain submodules, which contain sub-submodules, and so on.

MATH>Computer Science>Software>Programming>Architecture>Framework

framework in software

Abstract-class sets {framework} can work together in reusable designs.

request-response framework

Frameworks {request-response framework} can be integration services that send service requests to technical services, applications, and business-model instances and receive responses. Service-request {request, computer} execution environment can include required input. Service-response {response, computer} execution environment can include output.

services framework

Frameworks {services framework} can define and specify service uses.

MATH>Computer Science>Software>Programming>Architecture>Layers

software layers

System components {software layers} can be for applications, technical services, integration services, business services, and data services.

application layer

Software layers {application layer} can include installers and user interfaces and use business and technical services, through integration services. Interfaces are for navigating, querying, calculating, displaying, and reporting.

business layer

Software layers {business layer} can include instrument controls, analysis tools, quality-control modules, parameter settings, queries, and business objects. Models {business model} can be business-object content, behavior, rules, and interactions. Data objects {business object} are in systems. Service groups {business service} access and control shared domain-specific service interfaces, such as instrument controls and data.

data layer

Software layers {data layer} can include auditing, ID, password, authentication, persistence, and relational database. Implementation-model subset {data model} describes persistent procedure, trigger, and constraint logical and physical data representation.

domain layer

Software layers {domain layer} can contain modules working with data objects. Business-model subsets {domain model} can have business types.

integration layer
Software layers {integration layer} can include request-response, instrument, workflow, directory, and messaging services. A service group {integration service} includes request-response framework, architectural framework, and services that access and control shared services.

presentation layer
Software layers {presentation layer} can contain user interfaces.

MATH>Computer Science>Software>Programming>Architecture>Services

software services
Querying, versioning, or persisting data {system service} {software services} happens upon request for service or system function. Software components {service provider} provide service functions, such as for instrument runtime or messaging. System-architecture primary services allow other-service development.

Java Database Connectivity
Java class and component-library APIs {Java Database Connectivity} (JDBC) can allow Java applications to communicate with database server, outside database software.

Java Message Service
Services {Java Message Service} (JMS) can publish and listen for messages.

Java Naming Directory Interface
Java extensions {Java Naming and Directory Interface} (JNDI) can provide API for accessing directory and naming services, such as LDAP, Novell Netware NDS, CORBA Naming Service, and naming services provided by EJB servers. EJB servers organize beans into directory structure and provide JNDI driver service provider for accessing directory structure.

naming service
JNDI database {naming service} contains named objects, with no attributes.

technical service
Infrastructure services {technical service} can be for applications and business services: logging, security, exceptions, and system management.

MATH>Computer Science>Software>Programming>Languages

programming language
Computer instructions have exactly defined and limited vocabulary, syntax, and grammar {formal language, programming} {programming language}| {computer language}, with no ambiguities or connotations.
parts
Programming languages have alphabet {character set} that makes numbers and words in formats. They have mathematical functions, logical operations, and string manipulations {operator, programming}. They have processing instructions {executable}. They have sentences {declaration, programming} {non-executable declaration} that name, set up arrays, or create symbols.
levels
Machine uses machine-dependent binary code {machine language}. Assembler uses machine-dependent three-letter codes {assembly language} for instructions and addresses and has one statement for each instruction. People use higher-level languages {compiler language} {high-level language}, with one statement for several instructions, such as BASIC, FORTRAN, COBOL, UNIX, and DOS.

ALGOL language
early procedural computer language {ALGOL language}.

BASIC language

Simple basic-function languages {BASIC language} can be for interactive use.

COBOL language

early procedural computer language {COBOL language}.

Java language

Computer languages {Java language} can be object oriented. Java has classes with methods. A main static method may be first. Methods use "get" and "put" for getting and setting values, "is" or "has" for boolean test functions, and action verbs for methods that change state or take action in other ways. Methods should proceed from public to protected to private. Within each group, methods should start with constructors, then instance methods, then static methods. Java has local and global class member instance variables and static member variables. Java has constants. Create output using System.out.println. The entire system has to be layered and buildable in a single pass. A script defines the order in which the packages are built. Java 2 Enterprise Edition (J2EE) defines application servers for Enterprise Java Beans (EJB), Servlets, and Java Server Pages (JSP), to make web-enabled applications.

LISP language

Interactive computer languages {LISP language} can be for symbol manipulation. Symbols can be numbers, words {atom, LISP}, or groups {list, LISP}. Data and operations are both lists {list structure}. Programs use, modify, and recognize lists. List addresses contain two pointers, one to current list value and other to next list item. LISP can perform all string operations and compare lists and operations. Lists can have any number of dimensions.

SQL language

Relational database queries use a language {SQL language} {Structured Query Language}. Make variable names no more than 31 characters. Declare cursors explicitly. Primary cursors remain open. Secondary cursors close after fetching all rows.

table

In a database, each data structure is a table, with columns and rows. The rows are records, a set of information about one item, and the columns are each attributes of the item.

datatype

Each column has entries all in one datatype. The datatypes are integer, floating point number, fixed length field of n characters, variable length field of characters, bit, binary fixed length field of n characters, money, and date and time. A "null" entry means an unknown entry, not blank or zero.

queries

select column, column, ... from table - retrieves data from columns in the table and displays in the order specified.

select column from table where phrase - phrase sets conditions for selection. The phrase may use symbols such as =, >, <, >=, <=, !=, !>, and !< and may use connectors such as and, or, not, between m and n, is null, is not null, in (...), not in (...), and LIKE ' '. The symbols that can be used in LIKE are % for any string of 0 or more characters, _ for anyone character, [ ] for any single character in a range or set, [^] for any single character not in a range or set.

function

select function from table - runs a function, such as "getdate ()".

order

select column from table order by column - sorts in ascending order.

select column from table order by column desc - sorts in descending order.

group

select column from table group by column, column, ... - makes nested groups.

select column from table group by column having phrase - phrase sets conditions for grouping.

operator

select operator(column) - operators check numbers of entries: count(column), max(column), min(column), sum(column), avg(column).

insert

insert table (column, column, ...) (column-1-name datatype, ...) values ('x', 'y') - inserts rows into table at the columns specified using the values in order.

update

update table set column, column, ... from table where phrase - modifies one table by updating its rows.

delete
delete table from table where phrase - deletes rows from table.
Exception
EXCEPTION traps errors. Some named exceptions are ZERO_DIVIDE, NO_DATA_FOUND, TOO_MANY_ROWS, VALUE_ERROR, OTHERS (for all errors). For example, EXCEPTION WHEN ZERO_DIVIDE THEN RESULT1 = NULL;. It has no continue.

MATH>Computer Science>Software>Programming>Techniques

code access
Use all function arguments. Be able to access all code {code access}.

code comments
Add comments {code comments} for logical units. Add comments for modifications. Add inline comment after final brace to identify original brace.

conditional statement in software
Use IF-THEN-ELSE {conditional statement, software}, rather than multiple IFs. Use compound IFs, rather than nested IFs. Use braces for nested IF blocks. Always use ELSE after ELSE IF. Enter NULL statement ";" in IF block with no action. Use Switch statements, rather than IF-THEN-ELSE, if possible. Use default Switch at end, containing at least break statement.

constant in software
Use symbolic constants {constant, software}, with no hard-coded numbers.

data type in software
Do not use mixed data types {data type} in expressions. Do not use machine-dependent data representations.

dynamic memory
Use dynamic memory {dynamic memory, computer}, rather than allocating fixed storage. Verify that memory gave information. Free all memory structures no longer used. Allocate memory to structures to keep before structures to free, to avoid memory leaks.

error message in software
Use parameters for error messages {error message} and error numbers, not many hard-coded messages.

file header
Put file name, author, creation date, description, purpose, and modification history in file header {file header, information}.

function header
Put function name, author, creation date, short description, usage, possible return codes, and modification history in function header {function header}.

function return
Return success or failure status {function return} {return, function} to caller. Check status in caller. Store function-return codes in caller for future reference. Do not test function logical value directly. Declare return "void" for functions that do not return value. Declare argument list "void" for functions that do not use parameters.

GOTO command
Do not use GOTO statements {GOTO command}.

header file
Put definitions, global-variable declarations, and function prototypes in header files {header file}. Use "include" statements. For static functions used by one module, put prototype at module top.

input in software

Check input values {input, software} for range and reasonableness.

looping in software

Do not reinitialize index variables in loops {looping, software}. Do not use non-loop-dependent statements in loop bodies.

module files

Put modules in separate files {module files}. Put source code for C++ classes in .CXX files. Put class definitions in .HXX files. Use only logically related functions. Use one entry and exit point for each function. Exit program only through "main" command.

object names in software

Make object names {object names, software} descriptive and globally unique. Use suffix for object types, like libraries. For functions, use lowercase or mixed case. For variables, use lowercase or mixed case. For constants, use uppercase.

operating system services

Minimize operating-system services {operating system services}. Put operating-system services in "system" library, not in high-level code.

readability in software

Separate variables and operators {readability, software}. Write one statement per line, using consistent indentation for logical structure. Place lead brace on same line as statement, or place on next line in same column as statement beginning. Put end brace in same column as statement beginning. Indicate conditional statement using inline comment after end brace.

standard usage in software

Use only standard programming language {standard usage, software}.

variable declaration

Use one line for each variable declaration {variable declaration}. Minimize number of global variables and external variables by passing arguments by value or reference. Declare arrays and do not use pointers except for dynamically allocated memory. Use structures for related data elements, to treat them as single objects.

WHILE command

Do not exit from WHILE-statement body {WHILE command}.

MATH>Computer Science>Software>Programming>Techniques>Objects

encapsulation

Only methods can access objects {encapsulation, programming}.

inheritance

You can make a child class with all the members and methods of a parent class, and then add extra members and methods {inheritance, programming}.

polymorphism programming

If classes share common methods, the same methods can apply to all the classes {polymorphism, programming}.

Liskov Substitution Principle

You can use a child-class instance if you can use a parent-class instance {Liskov Substitution Principle}.

MATH>Computer Science>Software>Programming>Kinds

programming types

Programs can be procedure-oriented, non-procedural, problem-oriented, application-oriented, special purpose, problem defining, problem describing, or problem solving {programming types}.

functional programming

Programming {functional programming} can change input to output by performing logical relations on input. Sequence and location are not important, because input calls functions, not registers.

procedural programming

Programming {procedural programming} can perform actions on register states to get new register states. Actions must be in exact sequence at one location. Modules are command sets that perform one process or procedure {process box}, such as moving values, performing logical AND, executing if-then-else statement, or iterating through do-while or perform-until statement. Modules can be scripts. BASIC depends on procedures or scripts and is useful for small systems and rapid development.

satisfiability solver

Software can start with objects and find arrangements that satisfy rule {satisfiability solver} (SAT), in reasonable time.

structured programming

Programming {structured programming} can use independent modules. Programming languages can use objects {object-oriented programming} {component architecture}, rather than procedures. SVL and Perl have both objects and procedures. C++ and Java are almost all object-oriented, can code larger systems, and can require more development time.

Rather than procedures, structured programming languages, such as Java, use independent modules {class} about data types (abstract objects) more complex than basic data types such as integers, floating-point numbers, strings, dates, and arrays. Class groups {Java package} are about one system part, such as file control, database, output, input, or flow control. Software systems are package groups.

Objects model data structures such as windows, files, tables, databases, database query results, and images. Data structures have parameters and constant and/or variable properties.

Classes have procedures {methods} about data structures:

- Make {constructor} a specific instance of an abstract object with specific property values.
- Make an instance {instance method} of an object with a procedure and a returned value.
- Open and close windows.
- Open file, write to file, read from file, and close file.
- Get table-cell value and set table-cell value.
- Change object state or property.
- Indicate if statement about object state or property is true or false.

Methods can have parameters and return result values. A **main** static method starts the program and has a procedure that calls objects.

Methods, classes, and packages can call methods, classes, and packages. Classes and packages can share global constants and variables and can modify global variables.

variables

Programs can use global variables in different ways, or different functions can modify global variables {coupling, Java}. Constants can be global.

unique

Functions, methods, classes, and packages have only one copy, called by other functions, methods, classes, or packages.

comments

Packages, classes, functions, and variables can have comments.

format

Commands at same level indent, typically four spaces, to same position. Line lengths are less than 80 characters. Open braces are at line ends, with their close braces on one separate line lined up with opening-line first character {Microsoft style}.

naming

Object names typically have capitalized syllables or words, as in ElecModelWrapper. High-level objects are nouns, as in ElecModelWrapper. Action names start with verbs, as in writeThisFile. Queries start with "get", as in getString. Information collection starts with "put", as in putString. True or false indicators, with boolean datatype, start with "is", as in isConnected. Names about same thing start the same, as in SysProps and SysVars. Typical prefixes are c for constants, g for globals, m for member variables, s for statics, v for variables, and p for parameters. Names are longer than two characters and use letters rather than numbers.

MATH>Computer Science>Software>Speech Recognition

speech recognition

Put sounds in intervals, measure energies, identify words, and parse sentences to recognize speech {speech recognition}. Segment acoustic waveform into ten-millisecond intervals {segment, sound} {sound segment}.

energy

In intervals, measure several frequency-band energies from 200 Hertz to 6000 Hertz. Linear predictive coding coefficients, zero-crossing rates, glottal frequency, and total energy are measures.

word

Energy sustained over several intervals indicates word. Interval number is word duration. Compressing longer words and elongating shorter words can standardize word duration.

pause

Low energy in one or more intervals indicates pause between words.

identification

Compare word frequency-band energy spectrum to stored word templates to identify word.

parsing

Use word syntactic and semantic information to parse sentence.

language understanding

Systems {language understanding system} can detect strings using features and then classify strings and substrings by syntax.

MATH>Computer Science>System Analysis

systems analysis

Systems analysis {systems analysis} {system analysis} {system design} tracks inputs, processing, and outputs.

theories

Artificial intelligence, artificial life, catastrophe theory, chaos theory, complexity theory, computational complexity theory, cybernetics, dynamical systems theory, evolution theories, experimental mathematics, fractal geometry, general systems theory, nanotechnology, scientific computing, self-organization, and statistical mechanics are theories about systems and can be analysis tools.

changing

To change system, change weakest part first. Change only one step or thing, then test system behavior [Kampis, 1991].

matrix

Matrices indicate part-pair relation and interaction strength. Hypermatrices can show secondary interactions and synergisms.

iterations

Systems can repeat processes, to converge on solutions. Fast systems, algorithms, and processes are unlikely to use loops and iterations.

goal-driven system

Goals are ideal states to approach or bad states to avoid. Programs measure progress toward goal after each step. Systems {goal-driven system} can try to minimize difference between current situation {origin, state} and desired situation {destination, state}. Systems can try to maximize difference between current situation and undesired situation. Goal-driven systems typically have subgoals and subsystems that lead to main goal.

To reach goal, systems usually use association and recall, rather than logic or statistics.

interactive system

Systems {interactive system} can be complex, chaotic, and unpredictable. They can self-organize, because interactions assume stable patterns. Other parts can assume damaged-part or overloaded-part functions.

operations research
Data flows from input, to data capture, to data processing, to output, to output distribution {operations research}|. At each step, there must be error prevention, error detection, error correction, security, backup, and controlled access.

categoricity
Theories can describe pure structure categories {categoricity}, with no reference to physical objects or events. Systems with that structure category are isomorphic to all other structures with that structure category.

MATH>Computer Science>System Analysis>Definition

system by definition
Independent object and relation sets {system, definition} can have limited and special interactions with outside.
outside
Systems {open system} can interact with outside world and take in or give out negentropy. Open systems can reach same state in different ways. Systems {closed system, analysis} can be separate from outside world.
objects and properties
System objects and events have properties, which have values. Systems, codes, or machines have one or more independent elements and element types. For example, elements can be numbers, letters, words, picture elements, sounds, tastes, smells, pressures, emotions, and signals. Elements can have sequences, patterns, or structures. Element properties can have value ranges.
relations
Relations determine how property values change. Real systems have forces and transfer energy. Interactions among system parts define system functions, goals, or dynamics. Parts can exchange data to provide input to one part from other-part output {communication, system}. Parts can combine to perform subfunctions {cooperation, system}. Parts can increase other-part activities or amounts by positive feedback. Parts can decrease other-part activities or amounts by negative feedback. Series can use output as input in chain reactions.
states
At all times, system objects have property values.
purpose
System typically describe action or how to perform action. Systems describe how to manage data. Goals are ideal states, in which variables have optimum values. Systems can try to maximize, minimize, or optimize variable values.
process
Systems select input data, represent data, process data, and output data. Processing stores, recalls, connects, and compares data. Given inputs, outputs have probability. Systems approach, avoid, use, control, or act on subsystems or environment. Systems select from choices. Systems recognize objects or events. Systems respond to input and/or state. Systems have stored facts and actions to use as resources.
problems
Systems with too many interactions are fragile. Systems with too many similar interactions are too redundant to adapt to novel situations. Systems must have enough time to process information to avoid mistakes and act correctly [Lévi-Strauss, 1966] [Lévi-Strauss, 1969] [Lévi-Strauss, 1985].

formal system
Information processing produces output from input using functions and symbols {formal system, analysis}, without using intentions or intentionality.

metron
Elements can move or change independently. Independent elements have degrees of freedom {logon}. Elements have weights or probabilities, which form metrics {metron}.

operator in system
Systems have one or more operations or relations {operator, system} among elements. Operators can start with existing elements and make same element types or make new element types.
types

Operators can be negating {unary operator}; adding, multiplying, linking, associating, and surrounding {binary operator}; association, chord, and color making {tertiary operator}; or object making and scene making {n-ary operator}.

relations

Familial relations can be parent-child, sister-sister, cousin-cousin, and so on. Object relations can be whole-part. Spatial relations can be above-below, left-right, front-back, inside-outside, and so on. Temporal relations can be before-after.

MATH>Computer Science>System Analysis>Outputs

equivalent output

Systems can have the same output {equivalent output} as other systems. First, identifier module samples system input and output to identify system parameters that model system. Second-system control module uses system parameters that model first system and feedback signals from second system to add control signal to second-system output.

identification problem

Two different systems can have same outputs for same inputs, so people cannot know machine nature from inputs and outputs alone {identification problem}.

MATH>Computer Science>System Analysis>Model

model

Ordered sets {model, system} {software model}, of various elements, can represent object or process properties. Models simulate positions, motions, times, momenta, energies, and interactions. Models can be physical or mathematical but have actual physical actions. Model elements have or represent points, lines, angles, surfaces, and/or solids, so models have geometric relations. Typically, model shapes are similar to physical shapes. Model and physical metrics are proportional. Model motions represent physical motions. Typically, model motions are similar to physical motions. Function models have moving parts that change positions and motions to represent events and transfers among states.

purposes

Constructing models can find what is important and not important, sharpen definitions and categories, identify procedure steps, identify structure parts, test interactions, sharpen boundaries, make categories, and uncover new relations, meanings, and properties. Constructing models adds information that causes understanding.

Models can represent system views. Models try to include constancies in real system and relations among elements. Models can simulate one description level or one input/output signal class. Models can measure information flows and constraints. Models can test knowledge or predictions and can reveal new object or process knowledge.

symbol grounding

Models can have parts and functions that relate to physical parts and functions {symbol grounding, model}.

black box

Systems {black box}| can have only inputs and outputs, with no process explanation.

computation theory

Models {computation theory} must specify what to compute.

formal model

Models {formal model} can put relations and components into symbols. Solutions come from mathematical analysis.

scale model

Models {scale model}| {replica} can copy larger-object shapes.

simulation on computer

Models {simulation, computer} can put relations and components into symbols. Computers find iterated or statistical solutions [Pellionisz and Llinas, 1982].

MATH>Computer Science>System Analysis>Learning

learning in systems

Systems can learn {learning, machine}, if parts or part relations can alter. Learning allows new states and/or trajectories. Learning requires mechanisms that can change system relations or rules. Learning requires input information. Separate evaluation function indicates success or failure in performing system function.

competitive learning

Networks {competitive learning} can use units that inhibit nearby units, creating competition among units.

constraint satisfaction

Networks can adjust connection strengths among nodes {constraint satisfaction}|, using feedforward and feedback, to find complete and consistent input-property interpretations.

conspiracy effect

If new pattern is similar to training patterns, new pattern enhances all network nodes {conspiracy effect}. New pattern unrelated to training patterns degrades training.

MATH>Computer Science>System Analysis>Network

association graph

Networks can link nodes by associations {association graph}, with or without scaling.

connectedness in networks

Networks can have only one or two inputs to each node {sparsely connected}. Networks can have four or more inputs to each node {densely connected} [Kanerva, 1988] {connectedness}.

feedforward network

Networks {feedforward network} can have interconnected units that adjust connection strengths to store processes or representations. Input-layer nodes receive intensities. Middle-layer nodes connect to all input nodes and to all output nodes. Nodes at same level do not interact. Output layer indicates answers by output pattern at nodes. Output from hidden units logarithmically relates to input.

McCulloch-Pitts neuron

Neuron models {McCulloch-Pitts neuron} can use linear threshold logic units. It strengthens synapse if input fails to fire neuron when expected, or weakens synapse if input fires neuron when not expected. Networks can use McCulloch-Pitts neurons to store representations, match input to representations, and send output.

pandemonium network

Networks {pandemonium network} {contention scheduling network} {winner-take-all network} depend on competition among processes {demon}, until only one process is still active. Representations try to inhibit all other representations. The strongest inhibits all others, and program selects it.

Perceptron

Network input-output devices {Perceptron} can alter connections or connection strengths by adjusting weights based on input, using feedback that output was correct or incorrect compared to ideal output {Perceptron learning rule}. Ideal output is one pattern or separable linear patterns. In initial learning period, Perceptrons adjust weights. In later test period, Perceptrons send more or less correct output for inputs.

random graph

Systems {random graph} can have nodes and edges with no organization and no order.

clustering

If edge number is smaller than half node number, so edge-to-node ratio is less than 0.5, few nodes cluster, largest cluster is small, and most nodes do not connect to other nodes. Connection growth rate is greatest as edge-to-node ratio increases from 0.5 to 0.6. If edge-to-node ratio equals 0.6 system has phase transition, most nodes cluster, largest cluster is big, and most nodes connect to other nodes. After that, growth slows, because most nodes have connections already.

node number

If node number is small, phase transition has wider edge-to-node ratio. If node number is large, phase transition has smaller ratio.

reaction graph

Graphs {reaction graph} can have nodes that are compounds, such as polymers, and connectors that are reactions.

nodes

Polymers have different lengths. Polymers are products of reactant polymers. Compounds can supply energy to make polymers. Required compounds are polymer units and do not have input reaction paths. Some compounds are never reactants and have no output reaction paths.

connectors

Connectors lead from reactant nodes to product nodes {reaction path}. Reactions can lengthen or shorten polymers. Lengthening polymers requires energy. Shortening polymers releases energy.

process

If existing polymers can make more-complex polymers, number of compound nodes increases and number of reactions increases more. If many different reactions make and break polymers, reaction-to-compound ratio increases exponentially.

catalyst

Reactions can have input catalysts that increase reaction rate but that reaction does not consume. Reaction-graph subsets {catalyzed reaction subgraph} can have only catalyzed reactions.

autocatalytic

Systems {autocatalytic system, reaction graph} can have reactions whose products are reactants and increase rates of reactions that depend on concentration. Autocatalytic systems consume original reactants quickly and end quickly. Catalyzed reaction subgraphs {autocatalyzed subset} are self-catalyzing if all compounds are either food or are catalysis products.

If existing polymers can make more-complex polymers, a small percentage of new polymers can be catalysts. When catalyzed-reaction number becomes more than polymer number, phase transition goes to autocatalytic system.

If autocatalytic systems have food and energy compounds, number of different polymers can increase. If system can double, probably requiring specialized catalysts and molecules, it has reproduced itself {self-reproducing}.

critical

Reaction graphs can have chain reactions {criticality, reaction}. Chain reactions can make same molecules {subcriticality}, so number increases exponentially. Chain reactions can make new molecule types {supracriticality}, so chain reactions make exponentially more types. If different-object-type number increases, supracritical behavior increases. If reaction-catalysis probability increases, supracritical behavior increases.

transition network

Networks {transition network} can represent objects as nodes and conjunctions between objects as arcs.

purposes

Transition networks can model binary object conjunctions by AND operations. Transition networks can represent processes. Transition networks do not model object quantities. Transition networks do not model disjunctions, such as inclusive OR.

transition

System states have specific nodes and arcs. Transition from one state to another state has probability. Transitions have directions. There can be final state. Transitions can depend on time, properties, and previous transitions.

comparison

Transition networks relate to algorithms and grammars.

MATH>Computer Science>System Analysis>Network>Boolean

Boolean network

Networks {Boolean network} can have nodes that are either on 1 or off 0. Inputs from other nodes determine node state. Boolean rules can make values 0 or 1 equally likely, make value 0 certain, make value 1 certain, or make any probability. If average Boolean rule makes value 0 or 1 almost certain, system is stable. If average value makes 0 and 1 equally likely, system is unstable. At one probability, system switches rapidly from order to chaos.

canalyzing

Boolean networks {canalyzing Boolean function} can have input that determines node output. For example, OR has two inputs and is canalyzing, because if either input is 1, output is 1. EXCLUSIVE OR is not canalyzing, because inputs depend on each other. If input number is two, 14 of 16 possible Boolean functions are canalyzing. EXCLUSIVE OR and IF AND ONLY IF are non-canalyzing. For Boolean functions with more than two inputs, few are canalyzing. Canalyzing functions have fewer interactions and so are simpler.

MATH>Computer Science>System Analysis>Network>Neural Net

neural network

Interconnected units {neural network}| can adjust connection strengths to model processes or representations.

Each input-layer unit sends its signal intensity to all middle-layer units, which weight each input.

Each middle-layer unit sends its signal intensity to all output-layer units, which weight each input.

The system can use feedback [Hinton, 1992], feed-forward, and/or human intervention to adjust weights (connection strengths).

- To calculate adjusted weight W' using feedback, subtract constant C times partial derivative D of error function e from original weight W: W' = W - C * D(e). The program or programmer can calculate constant C and error function e.
- Alternatively, to calculate adjusted weight W' using feedback, add original weight W and constant C times the difference of the current amount c and estimated true amount t: W' = W + C * (c - t). The program or programmer can calculate constant C and estimated t.
- Widrow-Huff procedure uses f(s) = s: W' = W + c * (d - f) * X, where d is value and X is factor.
- Generalized delta procedure uses f(s) = 1 / (1 + e^-s): W' = W + c * (d - f) * f(1 - f) * X.

Input patterns and output patterns are vectors, so neural networks transform vectors (and so are like tensors). Computation can be serial or parallel (parallel processing).

Note: Units within a layer typically have no connections [Arbib, 2003].

Output units can represent one of the possible input patterns. For example, if the system has 26 output units to detect the 26 alphabet letters, for input pattern A, its output unit is on, and the other 25 output units are off.

Output unit values can represent one of the possible input patterns. For example, if the system has 1 output unit to detect the 26 alphabet letters, for input pattern A, output unit value is 1, and for input pattern Z, output unit value is 26.

The output pattern of the output layer can represent one of the possible input patterns. For example, if the system has 5 output units to detect the 26 alphabet letters, for input pattern A, the output pattern is binary number 00001 = decimal number 1, where 0 is off, 1 is on, and the code for A is 1, code for B is 2, and so on. For input pattern Z, the output pattern is binary number 11010 = decimal number 26.

Output-pattern values can represent one of the possible input patterns. For example, if the system has 2 output units to detect the 26 alphabet letters, for input pattern A, output-pattern value is 01, and for input pattern Z, output-pattern value is 26.

For an analog system, the output pattern of the output layer can resemble an input pattern. For example, to detect the 26 alphabet letters, the system can use 25 input units and 25 output units. For input pattern A, the output pattern resembles A. Units can have continuous values for different intensities.

uses

Neural networks can model processes or representations.

Large neural networks can recognize more than one pattern and distinguish between them.

They can detect pattern unions and intersections. For example, they can recognize words.

Neural networks can recognize patterns similar to the target pattern, so neural networks can generalize to a category. For example, neural networks can recognize the letter T in various fonts.

Because neural networks have many units, if some units fail, pattern recognition can still work.

Neural networks can use many different functions, so neural networks can model most processes and representations. For example, Gabor functions can represent different neuron types, so neural networks can model brain processes and representations.

Neural networks can use two middle layers, in which recurrent pathways between first and second middle layer further refine processing.

vectors

Input patterns and output patterns are vectors (a, b, c, ...), so neural networks transform vectors and so are like tensors.

feedforward
Neural networks use feed-forward parallel processing.
types: non-adaptive
Hopfield nets do not learn and are non-adaptive neural nets, which cannot model statistics.
types: adaptive
Adaptive neural nets can learn and can model statistical inference and data analysis. Hebbian learning can model principal=component analysis. Probabilistic neural nets can model kernel-discriminant analysis. Hamming net uses minimum distance.
types: adaptive with unsupervised learning
Unsupervised learning uses only internal learning, with no corrections from human modelers. Adaptive Resonance Theory requires no noise to learn and cannot model statistics. Linear-function models, such as Learning Matrix, Sparse Distributed Associative Memory, Fuzzy Associative Memory, and Counterpropagation, are feedforward nets with no hidden layer. Bidirectional Associative Memory uses feedback. Kohonen self-organizing maps and reinforcement learning can model Markov decision processes.
types: adaptive with supervised learning
Supervised learning uses internal learning and corrections from human modelers. Adaline, Madaline, Artmap, Backpropagation, Backpropagation through time, Boltzmann Machine, Brain-State-in-a-Box, Fuzzy Cognitive Map, General Regression Neural Network, Learning Vector Quantization, and Probabilistic Neural Network use feedforward. Perceptrons require no noise to learn and cannot model statistics. Kohonen nets for adaptive vector quantization can model K-means cluster analysis.
brains compared to neural networks
Brains and neural networks use parallel processing, can use recurrent processing, have many units (and so still work if units fail), have input and output vectors, use tensor processing, can generalize, can distinguish, and can use set union and intersection.

Brains use many same-layer neuron cross-connections, but neural networks do not need them because they add no processing power.

The neural-network input layer consists of cortical neuron-array registers that receive from retina and thalamus. Weighting of inputs to the middle layer depends on visual-system knowledge of information about the reference beam. The middle layer is neuron-array registers that store perceptual patterns and make coherent waves. The output layer is perceptions in mental space.

Neurons are not the input-layer, middle-layer, or output-layer units. Units are abstract registers that combine and integrate neurons to represent (complex) numbers. Input layer, middle layer, and output layer are not physical arrays but programmed arrays (in visual and association cortex).

Neural-network processing is not neural processing. Processing uses algorithms that calculate with the numbers in registers. Layers, units, and processing are abstract, not directly physical.

nerve net
Models {nerve net}| can simulate object-recognition neuron networks. Nerve nets assign weight to nodes. Node input intensity multiplies weight to give output. Vector sum of outputs over nodes is object representation.

Hopfield network
Networks {Hopfield network} can use content-addressable memory, with weighted features and feedback.

simple recurrent network
Unit sets {context layer, network} can receive a hidden-layer copy and then add back to hidden layer {simple recurrent network}.

tensor network theory
Geometrical neural nets {tensor network theory} can make space-time coordinate transformations [Pellionisz and Llinas, 1982].

MATH>Computer Science>System Analysis>Phase Space

phase space of system
Systems have things with features. System models {phase space, system}| can use abstract-space nodes to represent things and can use dimensions to represent features or factors. Nodes can be system states. Similar states are near each

other. Low-dimension systems have less information about nodes, because nodes have fewer factors. High-dimension systems have more node information. With more dimensions, phase-space-model predictability declines, information flows increase, and mixing increases.

plotting in phase space

Phase space can represent variable values over time {plotting}|. First dimension is for value at time t, second dimension is for value at time t + 1, and so on. For simple processes, phase spaces can have characteristic shapes.

projection in phase space

Phase-space points or trajectories can project onto two-dimensional surfaces or three-dimensional solids {projection, phase space}. New dimensions are phase-space-dimension composites.

return map

Phase spaces can have cross-sections {return map} {Poincaré map} {Poincaré section} of fewer dimensions. Phase-space points, nodes, and trajectories can project onto cross-section.

sparse population coding

Phase space can have only several widely separated nodes {sparse population coding}. Nodes can represent complex states. States are not similar to other states. Sparse population coding can be for pattern or object recognition [Kanerva, 1988].

MATH>Computer Science>System Analysis>Components

separability of modules

Systems can have independent modules {separability} {component separability} {module separability}. Modules are element groups. Connected-node groups can have few connections to rest of system. Modules are information-processing functions, which transfer input to output. Module outputs are input to other modules and same module. Module-relation pattern determines how modules communicate and affect each other {correspondence rule, module}.

component hierarchy

Many systems have lower-level and higher-level components {component hierarchy} [Pattee, 1973] [Pattee, 1995].

conflict of subsystems

Hierarchies always have conflicts {conflict, hierarchy} between subsystems at same level. Higher agents settle conflicts using laws, policies, communication, power, compromise, and mediation.

MATH>Computer Science>System Analysis>Controls

system controls

Subsystems {control subsystem} {system controls} at same level inhibit each other. Higher subsystems compare lower-subsystem outputs over time and space, to send summaries to even higher subsystems.

control problem

Inputs can control system performance {control problem}. Control can keep output near reference value {regulator problem}. Control can follow trajectories {tracking problem}. Good control methods use independent positive and negative signals with wavelength and amplitude ranges. Such control signals have close control, smooth response, and good sensitivity.

control express line

Paths {express line} {control express line}, from low-level to high-level nodes, suggest, find, activate, prime, or inhibit hypotheses, models, or patterns. Paths, from high-level to low-level nodes, find, activate, prime, or inhibit lower-level nodes or indexes, for searches.

MATH>Computer Science>System Analysis>Controls>Feedback

feedback mechanism

Some output {feedback} can be input to regulate output [Wiener, 1947]. Feedback can compensate for minor departures from output level.

loop

Feedback loop continuously measures output {indicator, control}, modifies input {executive organ, control}, connects indicator and executive organ {transmitter, control}, and supplies energy {motor}. System parts {feedback mechanism} can subtract actual from intended output and send more or less signal to decrease differences, using algorithms {identification algorithm}.

setting

Feedback refines behavior but does not set behavior level, which is set manually.

signal

Too-great signals {overcompensation} cause cycles, as output overshoots intended output. Too-small signals {undercompensation} are not enough to overcome noise or inaccuracies and so fail to return system to expected performance.

feedforward

Classification algorithms can use prototypes or templates {feedforward}. Classification results when stimulus parts closely match prototype or template parts.

feedforward mechanism

Input can regulate output {feedforward mechanism}, by sending signals based on system states and environment to enhance or initiate actions. Feedforward sets output level based on algorithm or system model. After sending feedforward signal, system sends no more signals for a time {refractory period, feedforward}, to allow time to check first-signal results.

examples

Feedforward classification algorithms include feature-based winner-take-all algorithms {Pandemonium algorithm}, feedforward neural nets using feedback during learning {backpropagation, feedforward}, tree-based classifiers, and parametric statistical modeling [Selfridge, 1970].

homeostasis in system

Mechanisms {homeostasis, system} can use feedback controls.

negative feedback

Feedback {negative feedback} can dampen responses to maintain goal level. Negative-feedback algorithms can use different comparator types. ON-OFF regulation uses a constant set point. Proportional regulation uses variable set points. For constant disturbances, integral regulation uses constant set points, but output change rate is proportional to input. Derivative regulation uses input-change rate, proportional to output [Kampis, 1991].

positive feedback

Feedback {positive feedback} can synchronize events and deliver maximum response quickly, good for behavior rituals.

MATH>Computer Science>System Analysis>Functions

system functions

Systems have functions {system functions}. Linear functions, such as polynomials or series functions, sum weighted harmonic frequencies or weighted power functions. Linear functions can be scalars or vectors.

cross-correlation function

Feature values can relate in direct or inverse proportion {cross-correlation function}.

curl function

Potentials are vectors. Field or potential cross product of del and f {curl, function} describes area density. Curl is vector-field rotation rate {circulation density}, with magnitude and direction. Curl is non-linear. Using no coordinates, curl is limit, as volume goes to zero, of surface integral over closed surface, of cross product of unit outward-normal vector and function, all divided by volume. Find region-boundary function value and then divide by region volume.

divergence function

Function dot product of del and f {divergence} describes vector-potential flow or flux. Divergences are scalars. Positive divergence means diverging. Negative divergence means converging. Divergence is limit, as volume goes to zero, of double integral over surface area of dot product of vector field and surface-area differential, all divided by volume. Divergence is limit, as volume goes to zero, of surface integral over closed surface of dot product of unit outward-normal vector and function, all divided by volume. Find region-boundary function value and then divide by region volume.

gradient function

Function del f {gradient, function} describes field or potential changes over space. Gradients are maximum field direction and magnitude vectors. Field or potential can be scalar or vector. Gradients are non-linear. Using no coordinates, gradient is limit, as volume goes to zero, of surface integral, over closed surface, of product of unit outward-normal vector and function. Find region-boundary function value and then divide by region volume.

radial basis function

Multivariate functions {radial basis function} (RBF) can be weighted sums of independent linear functions.

input

Inputs can be spatial coordinates, angles, line-segment lengths, colors, segment configurations, feature binocular disparities, or texture descriptions. Training uses input data points.

dimensions

Data points have distances from coordinate means: |x - t|, where x are data-point coordinate values, and t are coordinate means. Data typically has Gaussian distribution, which can be broad or narrow, along all dimensions. Dimension number is typically less than data-point number.

training

Training assigns weights to dimensions or factors.

test sum

Test data point has sum over all weighted dimensions. Comparing sum to input data-object sums can identify test object. For narrow Gaussian distributions, RBF is like lookup table, because test objects only match if input equals mean.

hyperbolic basis function

Functions {hyperbolic basis function} (HBF) can allow more flexibility than radial basis functions. Networks can express weighted function f*(x) = summation over i from 1 to N of c(i) * G(transpose of (x - t(i)) * transpose of W * W * (x - t(i))) + p(x), where x are data-point values, t(i) are means or centers, c(i) are weights or coefficients, p(x) sets f(x) = 0, G is Gaussian distribution, and W is square matrix. Norm has weights: transpose of (x - t(i)) * transpose of W * W * (x - t(i)).

MATH>Computer Science>System Analysis>Operation

system operation

Systems have operations {system operation}.

growth

Systems can start slow, grow, and then level off, in a sigmoid curve. Systems can have linear growth, at constant rate. Systems can have exponential growth, at rate that depends on current size. Systems can have second-order exponential growth, at rate that depends on current size squared. Systems can have differential growth, in which different parts have different growth rates.

newness

To have new behavior, complex systems require information from outside system.

outside influences

Complex systems protect against changes from outside by storing information about current system state, typically in templates, and replacing changed states or parts using that information.

prediction

Machines can predict variable value, then check actual value against predicted value, then adjust prediction method. Process requires information about current system state, including current variable value. Process requires information about factors affecting output value. Process requires information about goal value. Models can use varying inputs and model-parameter configurations to find output values and compare values to actual values. Machine or researcher can then adjust model.

randomness

Randomness directly relates to information. Random events can add information to system. If a random event with low probability happens, it has high information. New information masks non-random information. New information can make systems less stable.

Random nodes or events have different time and space distributions, with different noise types. Random nodes or events can have different distributions, depending on system size or time scale. Fractals can model random events with same distribution at different scales.

rates

Going from one state to another, such as returning from non-equilibrium state to equilibrium state, requires time. Systems can move from one state to another at different speeds. Systems can use internal and external rate controls. Systems have time scales.

representations

In complex systems, several configurations and parts can typically perform same function or hold same representation. Redundancy causes some paths and functions to be equivalent. The same output can result from different processing paths.

rules

Systems have rules, exceptions to rules, checks, and balances.

symbols

System only recognize and use inside symbols. Systems cannot use symbols from outside system. Outside symbols must translate into inside symbols. Systems have mechanisms or filters to receive outside data, extract outside symbols, and translate them to inside symbols.

time

Systems and networks require time controller to coordinate data flows.

critical relation

Relations {critical relation} can control relation series. Certain interactions can die out or reach limiting value {lock-in}. Relation can prevent or cause another relation {double bind}. Relation can diverge further, or increase intensity, in reaction to another relation {schismogenesis} [Bateson, 1972] [Bateson, 1979].

equilibrium transient

Systems can slowly reach new state, oscillate toward new state, or have other transient behavior {equilibrium transient}.

reliability of system

Systems are more reliable {reliability, system}| if they perform just one function, rather than several different functions. Systems are more reliable if they perform same function more than once. Systems are more reliable if they use random input samples, rather than only one input. Systems are more reliable if they can perform function subfunctions using non-random input, then combine sub-outputs with consistency and completeness to get whole output.

robustness

Robust systems have independent parts {robustness, system}|, rather than mutually dependent parts.

MATH>Computer Science>System Analysis>States

system state of system

At all instants, elements, objects, events, and properties have variable values. Variable-value sets are system states {state, system} {system state}. Systems can be in stable, unstable, or cyclic equilibrium or in stable steady state.

MATH>Computer Science>System Analysis>States>Trajectory

trajectory in systems

Systems have state sequences {trajectory, system}|. Trajectories are paths through phase space.

progression in systems

Systems can go to final state, repeat state, or never repeat, using constantly interfering states. System can have interactions that make states that interact to form temporary substates {progression}.

state transition graph
Tables {state transition graph} show all possible state transitions.

ballistic trajectory
System behaviors, once begun, can have no further regulation and so follow trajectory {ballistic trajectory}|.

ergodic process
States can have trajectories from all other states {ergodic process}|. Trajectories can have equal or unequal probabilities. Ergodic processes always have non-recurring loops, because system eventually returns to previous states. Because probabilistic, ergodic processes do not have recurrence.

reversible trajectory
If they have no attractors, have no loops, are deterministic, and have conservation, systems can run in reverse {reversible trajectory}.

MATH>Computer Science>System Analysis>States>Trajectory>Terminus

attractor in systems
Over time, systems tend to move to terminal state {attractor}|, such as constant flow. Trajectories near state tend toward state {attraction basin} {basin of attraction}. Attractors that have more trajectories going to them have higher probability.

catastrophe in system
Trajectories can result in state {catastrophe, system}| that stops trajectory or changes available states.

chaos trajectory
Trajectories can go from one state to another, with no trend toward final state {chaos, system}| [Gleick, 1987] [Lorenz, 1963].

dissipative system
Ordered non-equilibrium systems {dissipative system} can have steady state, because matter and/or energy flow through.

stability in systems
Stable systems {stability, system} have attractors with short state-cycle length and large attraction basins, so changing from one state to nearby states, or changing transition rule slightly, leaves system in same attraction basin. In unstable systems, if state or transition rule slightly changes, system changes to long state-cycle length {chaos} or changes attraction basins {catastrophe}. If network is sparsely connected, network tends to stay in same attraction basin. If network is densely connected, network tends to be chaotic or catastrophic.

state cycle
In deterministic complex systems, trajectories tend to go to repeated states {state cycle}| {recurrence} {oscillator, system}. Examples are pendulums, fluids, circuits, and lasers.
length
Cycles have number of steps {length, cycle}. If nodes have input from all other nodes, length is square root of number of states and is large, and number of attractors is number of nodes divided by natural-logarithm base e and is small. If nodes have two inputs, length is square root of number of nodes and is small, and number of attractors is number of nodes divided by natural-logarithm base e and is small.

MATH>Computer Science>System Analysis>Patterns

compact pattern
Patterns {compact pattern} can have only points, which connect horizontally or vertically to at least one other point.

value

Patterns have numerical values.

boundary

Unique patterns have unique boundaries and surfaces. If surface has value, pattern has value, and vice versa.

differences

Changing one pattern to another adds or subtracts one point.

number

Same number of points has same number of patterns. 1 point has 1 possible pattern. 2 points have 1 possible pattern. 3 points have 2 possible patterns. 4 points have 5 possible patterns. 5 points have 12 possible patterns. 6 points have 35 possible patterns. 7 points have 108 possible patterns. 8 points have 369 possible patterns. 9 points have 1285 possible patterns. 10 points have 4655 possible patterns. 11 points have 17072 possible patterns. 12 points have 63565 possible patterns. 13 points have 238299 possible patterns.

number: multiples

Multiple for each step is 1, 2.00, 2.50, 2.40, 2.92, 3.09, 3.41, 3.48, 3.62, 3.67, 3.72, and 3.75. Multiple for odd steps is 2, 6, 9, 11.89, 13.29, and 13.96. Multiple for even steps is 5, 7, 10.54, 12.61, and 13.66.

factoring

Pattern-number factors are 1*1, 2*1, 5*1, 3*2*2, 7*5, 3*3*3*2*2, 41*3*3, 257*5, 19*7*7*5, 97*11*2*2*2*2, 12713*5, and 79433*3.

point types

Patterns can have different point types, such as colors. For three point types, 1 point has 3 possible patterns. 2 points have 6 possible pattern. 3 points have 36 possible patterns. 4 points have 246 possible patterns. 5 points have 2115 possible patterns. Multiple for each step is 2.00, 6.00, 6.83, and 8.60. Multiple for odd steps is 12 or 58.75. Multiple for even steps is 41. Pattern-number factors are 3*1, 3*2, 3*2*2, 41*3*2, and 47*5*3*3.

transformations

Pattern translation, reflection, rotation, and inversion make same pattern.

diagonals

If points can connect diagonally, patterns are not fundamentally different, only less compact. Patterns with points connected diagonally can transform to connect only horizontally or vertically. For example, V is L rotated 45 degrees.

unique pattern representation

To be unique, pattern representations {unique pattern representation} must use pattern center and pairwise relations between points.

center

The center is x, y, and z coordinate means plus unit distance along higher dimension. The extra dimension avoids false equivalences that can happen if center lies near point.

vectors

Vectors go from center to pattern points. For vector pairs, find something like cross product. Calculate each pair only once, not again for different order. Ignore unit vectors. Square difference, such as x1*y2 - x2*y1, assigned to unit vectors. Sum squares. Alternatively, use sum square root.

value

Add all cross products. Resulting sum is unique for compact pattern.

program 1

In array, in first coordinate, 0 is for y-coordinate, and 1 is for x-coordinate. k = pattern size. p = pattern-point number. max(p) = k. m = pattern number. n(#, k, p, m) are point coordinates. v = 0. v1 = 0. v2 = 0.

For p = 1 To k + 1. v = v + n(0, k, p, m). v1 = v1 + n(1, k, p, m). Next p. v = v / (k + 1). v1 = v1 / (k + 1). For p = 1 To k. For p1 = p + 1 To k + 1. x1 = n(0, k, p, m) - v. x2 = n(1, k, p, m) - v1. x3 = 1. y1 = n(0, k, p1, m) - v. y2 = n(1, k, p1, m) - v1. y3 = 1. w2 = ((x2 * y3 - y2 * x3) ^ 2 + (x1 * y3 - y1 * x3) ^ 2 + (x1 * y2 - y1 * x2) ^ 2) ^ 0.5. v2 = v2 + w2. Next p1. Next p.

Program compares patterns rapidly.

eye

Eye can perform this computation, because it is pattern center, and all points are in front of it. Eye can compare patterns and judge distances.

program 2

Patterns with different colors or point types can have pattern representations. In first coordinate, 0 is for y-coordinate, 1 is for x-coordinate, and 2 is for color or point type. k = pattern size. p = pattern point number; max(p) = k. m = pattern number. n(#, k, p, m) are coordinates and point type. v = 0. v1 = 0. v2 = 0.

For p = 1 To k + 1. v = v + n(0, k, p, m). v1 = v1 + n(1, k, p, m). Next p. v = v / (k + 1). v1 = v1 / (k + 1). For p = 1 To k. For p1 = p + 1 To k + 1. x1 = n(0, k, p, m) - v. x2 = n(1, k, p, m) - v1. x3 = 1. y1 = n(0, k, p1, m) - v. y2 = n(1, k, p1, m) - v1. y3 = 1. w2 = ((n(2, k, p, m) + 1) * (n(2, k, p1, m) + 1) + (x2 * y3 - y2 * x3) ^ 2 + (x1 * y3 - y1 * x3) ^ 2 + (x1 * y2 - y1 * x2) ^ 2) ^ 0.5. v2 = v2 + w2. Next p1. Next p.

order group
Sets can have symbols in sequence {order group}, such as pattern or k-tuple.
subsets
The set can have subsets. Subsets are symbol sets in sequences and patterns.
group
All subsets form order groups. For example, pattern "acg" has subsets NULL, "a", "c", "g", "ac", "cg", and "acg". Order groups contain null set and pattern. If sets can be circular, the set can have subsets "ga", "cga", and "gac".
equivalence
Rules can be that patterns are equivalent over gaps and insertions, so "acg" = "ac gX". Gaps or insertion size or number can have restrictions.
alignment
Two patterns share largest subset. Two patterns share two largest equivalent subsets at optimum alignment.
process
To compare patterns, add or remove gaps and insertions from both patterns to find largest subset. If symbols are dimensions, spaces have maximum number of shared dimensions and minimum number of new dimensions.
index
Indexes are part of, and have position in, patterns. Pattern symbols have one or more indices. In pattern "acgta", symbol "a" is at position 1 and 5. Pattern subsets start at one or more indices. In pattern "acgta", subset "ac" starts at position 1.
union
Combining patterns results in new symbol sequences and patterns. Start with first pattern and add new symbols in sequence. Discard symbols that are the same in sequence, for example, "abc" and "ag" nets "abcg". It is like union of sets but with order in elements.

Combining is associative but not commutative. Null pattern combines with pattern to give same pattern. The same pattern combines with itself to give same pattern. Inverse pattern combines with pattern to give null pattern, but there can be no inverse pattern.
intersection
Finding largest aligned subset is like set intersection. Aligning "abc" and "ag" nets "a".

Aligning is associative and commutative. Null pattern aligns with pattern to give null pattern. The same pattern aligns with itself to give same pattern. Inverse pattern combines with pattern to give null pattern.
conversion
Natural or artificial objects, events, lines, surfaces, solids, n-dimensional figures, geometric points, figures, or images can be linear single-symbol series and so can be patterns. Patterns have order groups, and so all things can align. For example, letter "a" can stand for angle of 45 degrees and letter "L" can stand for angle of 90 degrees, so pattern "aLa" can stand for right triangle with two 45-degree angles.

Symbol sequences can transform into symbol group sequences. For example, pattern "acgta" has three-symbol subsets, "acg", "cgt", and "gta", rather than a five-symbol sequence. Subsets can align rather than single symbols.

Two objects or events can transform into linear RNA-base sequences. They can align by hybridization.
length vs. symbol number
Things can use patterns with few symbols and long sequences or shorter sequences with more symbols.
brain
Perhaps, brain can compare patterns using order groups.

MATH>Computer Science>Systems

algorithm generator
Cellular automata, mobile automata, Turing machine, substitution system, sequential-substitution system, tag system, cyclic-tag system, register machine, and symbolic system can generate algorithms {algorithm generator}.

artificial life
Programs {artificial life} can model reproduction, movements, or reactions.

contour map
Systems have positions {contour map}, at which variables are constant. Systems have contour series.

cybernetics
Feedback and feedforward controls systems {cybernetics}|. Systems have logical properties, control mechanisms, and communication patterns.

experimental mathematics
Running programs and analyzing results finds proofs and results {experimental mathematics} [Alexandrov et al., 1984] [Hersh, 1997].

fractal geometry
Nested patterns are self-similar {fractal geometry}.

General Systems Theory
Systems theory {General Systems Theory}| can use calculus, compartment theory, control theory, cybernetics, decision theory, factor analysis, game theory, graph theory, information theory, network theory, operations research, queuing theory, set theory, simplex methods, stochastic modeling, and equation systems [Bertalanffy, 1928] [Bertalanffy, 1968] [Thurstone, 1935].
relations
Networks have interacting nodes. Relations among system objects can be strong or weak, linear or non-linear, and hierarchical or otherwise. Relations can involve progressive differentiation, action triggers, feedback, and adaptations.
strategies
Systems can have growth/stability, competition/cooperation, centralization/decentralization, and specialization/generalization.

lambda calculus
Symbol strings can represent numbers and functions. Using functions on input function and data strings makes output function and data strings {lambda calculus}. Systems expressing lambda calculus make many new strings. Removing random, irregular, or long strings can keep number of strings constant, and system can evolve toward stable set of strings. One string can copy itself better than others and become more numerous than other strings {level-0 organization}. Removing strings that copy themselves causes one or more repeating string cycles {level-1 organization}.

leaking waterwheel
Lorenz systems are like waterwheel {leaking waterwheel} with leaking buckets that receive more water regularly.
low flow
Adding small drops slowly does not move waterwheel, because buckets leak equal or more water than they receive. Force is not enough to overcome frictional force.
high flow
With bigger drops or higher dripping rate, waterwheel turns steadily. With even bigger drops or even higher rate, waterwheel eventually spins so fast that buckets do not get as much water and buckets do not lose as much water each turn. Then waterwheel slows non-linearly. Spin can reverse.

limit cycle
Non-equilibrium systems can oscillate {limit cycle, system}, rather than achieve steady state.

linear programming
Linear equations {linear programming} can show how each factor {independent variable, linear programming} affects one output {dependent variable, linear programming}. Linear-equation sets can show how each factor affects each output.
purposes
Linear-equation sets can find optimum factor amounts and/or optimum output amounts.
factors

Factors are abstract-space dimensions. Factors can vary from zero to maximum amount. Factor amount marks position along factor spatial dimension.

simplex

In abstract space, connecting coordinates with lines, planes, and higher-dimension surfaces makes polyhedrons, centered on origin. Polyhedron vertices can be optimal or extreme solutions, showing how much factor to use {simplex method}. Extreme solutions can maximize output, maximize profit, maximize efficiency, and minimize cost.

methods

Analyzing extreme solutions uses marginal value. Optimization methods {sequential optimization method} can use slopes.

summation

Linear processing adds things of same kind and cannot model interactions between different things. Linear processing cannot transform data or reduce variable number, because sums have same dimensions as things added.

Lyapunov exponent

Numbers {Lyapunov exponent} can measure phase-space dimension extension, compression, or folding. If Lyapunov exponent is greater than zero, dimension extends. If Lyapunov exponent is less than zero, dimension compresses.

Minimum Description Length Principle

The most frequently used representations use shortest code {Minimum Description Length Principle}.

parallel processing

Systems can perform many similar processes simultaneously {parallel processing}|, using many independent processing paths. If process inputs or outputs go to other processes, system must have controls that coordinate signals in time and space. If parallel processes are redundant or can substitute for each other, system is more stable.

Post grammar

Processes {Post grammar} can replace input string with output string of same or different length. In simple Post grammars, short input strings have output strings listed in a lookup table {grammar table}. Complex Post grammars have long input and output strings, and rules determine output string substituted for input string. Rules are input strings. Rules can select input strings or string sequences. Rules can select or substitute strings deterministically or probabilistically. Post grammars can be equivalent to Turing machines and lambda calculi.

preference rule

Propositions {preference rule} {preference condition} can have conditionals and conclusion.

conditionals

Conditionals are positive or negative statements. Conditionals can link by conjunction or disjunction.

process

Systems can have preference-rule sequences. Systems test preference rules in sequence and perform first satisfied rule. If system satisfies conditions, preference rule applies, conclusion is true, and action starts. If any condition is not true, or if information is lacking for one or more conditionals, preference rule does not apply, and system tests next proposition. If no preference rule applies, system uses default action.

receiver-based optimization

All agents can communicate what they are doing to all other agents, and all agents decide what to do based on same goal or rule {receiver-based optimization}.

register machine

Two registers can have fixed lengths and two instructions {register machine}. 'increment' increases number in register by one. 'decrement-jump' decreases number in register by one and then goes to new program location.

scientific computing

Continuous equations can have discrete approximations {scientific computing}.

simulated annealing

Annealing {simulated annealing} can use heating, then cooling, then heating, then cooling, and so on. Heating makes molecular structure more random. Cooling makes molecular structure less random. In simulated annealing, system first minimizes or maximizes, then relaxes to farther-away position. It then repeats cycle. For example, system can minimize cost {cooling}, but occasionally explore higher-cost outcomes to go outside local minima {heating}, because neighboring outcomes have similar costs.

swarm intelligence

Ants and other social animals act and react based on signal communication, and net result is that whole colony has behavior patterns {swarm intelligence}. Ants leave pheromone trails. Pheromone evaporates over time. Ants taking shorter trails go from nest to food, and vice versa, more often, leaving more pheromone. Higher pheromone concentration attracts more ants. As they consume food, ants have less excitement and leave less pheromone, and fewer ants take that path, allowing ants to move more randomly and find other food sources farther away.

symbolic system

Strings transform according to rules {symbolic system}.

Turing machine

Abstract machines {Turing machine} [Herken, 1988] [Turing, 1937] run algorithms that have finite numbers of elements (typically 0 and 1) and instructions to execute (rule) and use only rational numbers, countable irrational numbers, and rational approximations to irrational numbers. Turing machines have a tape, a tape reader/writer with an internal state, and rules.

tape

The tape contains all input and receives all output.

The tape has only 0 or 1 at each of an infinite number of tape positions. The tape has infinitely many 0 positions and at least one 1 position.

The tape holds the input data, rules, and output.

Tape position combinations represent numbers, text, number-and-text separator symbol (comma), number-and-text termination symbol (blank), minus sign, plus sign, division sign, move right one square, move left one square, read, write, and stop. For example, the tape position combination ...001100... can represent the decimal number 3, the letter C, or the symbol minus sign. Coding is unique and unambiguous.

Combinations of tape position combinations represent numbers, words, sentences, mathematical formulas, dates, and arrays. Coding can represent all rational numbers and can represent approximations to countable irrational numbers, using minus sign, plus sign, and division sign. However, most irrational numbers are not computable by Turing machine, which never stops if something is uncountable.

The tape starts with all input data and instructions on one tape-reader side. Input data and instructions make a finite binary number. Instructions and input data depend on each other.

The tape ends with all output data on one tape-reader side. Output data makes a finite binary number. The other tape side includes intermediate calculations, modified input data, and rules.

Note: Turing machines have no need for more than one tape or tape reader, because multiple ones are always equivalent to one.

tape reader/writer

The tape reader/writer reads the 0 or 1 at the current tape position and writes the same or opposite symbol there (overwriting the previous symbol).

After writing, the tape reader/writer changes to the same or another internal state.

After writing, the tape reader/writer moves right or left one tape position. It can eventually move to any tape position, and go there any number of times, but does not have to go all positions or any one tape position.

internal state

The internal state is a combination of previously read symbols and is a finite binary number. Turing machines have a finite number of internal states, to which it can return any number of times, but does not have to reach all states or any one internal state.

The internal state and the currently read symbol determine the rule to apply after reading. Therefore, each internal state has a pair of rules, one for reading 0 and one for reading 1. Turing machines never reach an internal state that has no rules.

rules

For the current internal state and currently read symbol, rules (instructions) determine what the tape reader/writer does after reading the tape symbol: 1. Keep the same, or change to a different, internal state. AND 2. Change, or do not change, the symbol at the tape position. AND 3. Move tape one position to right or left. AND 4. Stop or do not stop.

Rules depend on input data. The number of rules is two times the number of internal states. Turing machines have a finite number of rules (because they have a finite number of internal states). Rules are finite binary numbers.

Rules can repeat any number of times. Turing machines do not have to use all rules or any one rule.

Turing machines must be able to get to at least one stop rule and so end operation.

output

After a stop rule runs, the tape has output data to the left or right of the final tape position. The output is a number, symbol, letter, word, phrase, sentence, date, or boolean yes or no (or an array of them). The tape's other side has intermediate calculations, modified input data, and the rules.

Turing machines must have at least one 1 in an output-side tape position.

operation

The tape reader/writer starts in an internal state at a 0 tape position on the right or left side of the input-data-and-rules tape positions.

The tape reader/writer reads the 0 at that tape position and, using the rule for the internal state and the 0 symbol, changes to the same or another internal state, prints the same or opposite symbol at the same position, moves the tape one position to right or left, and stops or does not stop.

Next, the tape reader/writer reads the 0 or 1 symbol at the new tape position and, using the rule for the current internal state and read symbol, changes to the same or another internal state, prints the same or opposite symbol at the same position, moves the tape one position to right or left, and stops or does not stop.

The tape reader/writer then continues reading symbols and following rules until it comes to a stop, which it must do or else it is not a valid Turing machine.

After the stop rule, the answer is to the left or right of final tape position. The other tape side has intermediate calculations, modified input, and rules.

Turing machines begin with a finite binary number, for inputs and rules, and produce a finite binary number, for output.

restrictions

Turing machine must have at least one rule that leads to STOP, must not move to non-existent internal state, must use and make only coded sequences of marked and blank squares, and must have at least one marked square on output side.

It is difficult to make input data and rules that reach STOP and so make Turing machines.

universal Turing machine

Some Turing machines (universal Turing machine), with special input data and rules on the tape, can imitate any Turing machine.

A simple cellular automaton and a special initial state can make a universal Turing machine.

computers and Turing machines

Computers are efficient universal Turing machines.

brains and Turing machines

Rather than one tape reader/writer, brains have many readers and writers acting continually and simultaneously. Upstream neurons automatically write to downstream neurons. Downstream neurons automatically read upstream neurons.

Rather than serial processing, neurons update simultaneously.

Rather than reading or writing one tape position at a time, brains continually and simultaneously read and write to all neurons.

Rather than separate tape positions, neurons interact.

Rather than a tape with an infinite number of positions, brains use a three-dimensional region with a finite number of locations.

Rather than two symbols to read, neuron synapses have synaptic strengths, which change continually.

Rather than two symbols to write, neurons send impulse rates, which change continually.

Rather than one internal state, brains have many internal states. Each neuron has a state, and each neuron set has a state. Brain, neuron set, and neuron internal states are complex and interact.

Rather than a small fixed set of rules, brains use many and more complex rules for changing state, reading, writing, moving, and stopping, and rules can interact and change.

Rather than stopping at a STOP rule, brains stop consciousness when they fall asleep but still process information.

Rather than the ability to perform any algorithm with a finite number of elements and instructions, brains may not be able to perform some algorithms, can perform non-algorithmic functions, and can use an infinite number of elements and instructions.

equivalences

Turing machines can compute partial recursive functions that use recursively enumerable element sets. Turing machines can approximate functions that are not partial recursive functions with partial recursive functions.

Because quantitative grammars involve only integers, Turing machines can be equivalent to quantitative grammars, Post grammar, and lambda calculus.

example 1: Turing Machine with One State, Two Rules, and One Marked Square

Rule 1: If State 1 and unmarked square, change to internal state 1, do not change the mark, move tape one square to right, and do not stop. Rule 2: If State 1 and marked square, change to internal state 1, do not change the mark, move tape one square to right, and stop. For example, see Figure 1 through Figure 5.

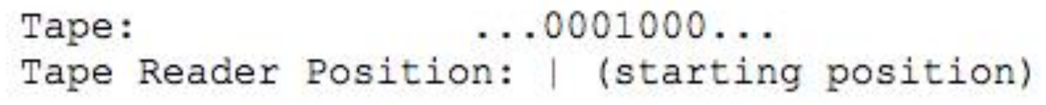
example 2: Turing Machine with Two States, Three Rules, and Two Marked Squares

Example Turing machine can calculate 1 + 1 = 2 or 01 + 01 = 10 in binary code. Infinite tape has square series that define rules. Then it has square series that define input: 0s, up to 1, followed by 1, and then 0s: ...00000110000... Reader starts to right of rules, reads rules, and ends to right of rules and left of the ...11..., in internal state 0. Rules for this Turing machine are as follows. If current state is 0, and 0 is read, move right to next input. If current state is 0, and 1 is read, move right to next input and add 1 to state. If current state is 1, and 1 is read, move left to next input and add 1 to state and stop. To tape-reader right is the answer 10.

example 3: Turing Machine with Two States, Four Rules, and Two Marked Squares

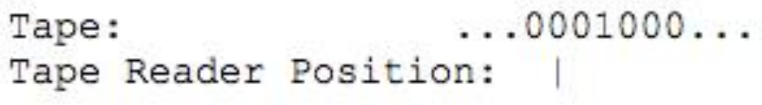
Example Turing machine can calculate 1 + 1 = 2 or 01 + 01 = 10 in binary code. Infinite tape has square series that define rules. Then it has square series that define input: 0s, up to 1, followed by 0, followed by 1, and then 0s: ...000001010000... Reader starts to right of rules, reads rules, and ends to right of rules and left of the ...101..., in internal state 0. Rules for this Turing machine are as follows. If current state is 0, and 0 is read, move right to next input. If current state is 0, and 1 is read, move right to next input and add 1 to state. If current state is 1, and 0 is read, move right to next input. If current state is 1, and 1 is read, move left to next input and add 1 to state and stop. To tape-reader right is the answer 10.

Figure 1



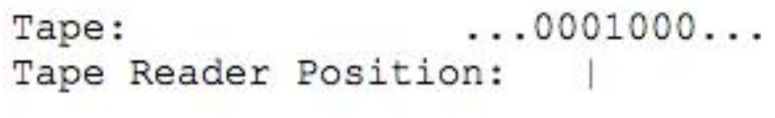
```
Tape:                    ...0001000...
Tape Reader Position: |  (starting position)
```

Figure 2



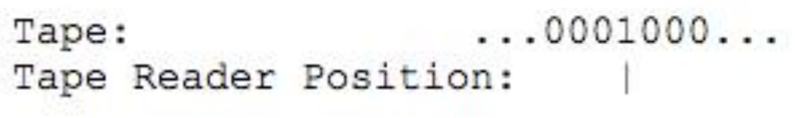
```
Tape:                   ...0001000...
Tape Reader Position:   |
```

Figure 3



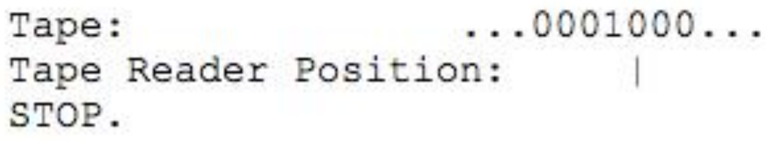
```
Tape:              ...0001000...
Tape Reader Position:   |
```

Figure 4



```
Tape:                  ...0001000...
Tape Reader Position:      |
```

Figure 5



```
Tape:                    ...0001000...
Tape Reader Position:         |
STOP.
```

MATH>Computer Science>Systems>Automaton

cellular automaton

Cells can update state at each step according to current state and neighboring-cell states {cellular automaton}|. Automata can have tapes with more than one dimension, with Turing machines at tape segments {node, tape segment}. Cells can have only one state and interact only with neighbors. Combining state and neighbor states determines cell next state [Wolfram, 2002].

mobile automaton

For fixed cell number, one cell can have one state and update at each step, and system can move to another cell {mobile automaton}, according to neighboring cell states.

MATH>Computer Science>Systems>Convection

convection in systems

Heating fluid at container bottom results in rising and swirling motion {convection, non-linear} {non-linear convection}.

warm

If hot fluid rises very slowly into cooler fluid, it can transfer heat by contact faster than by convection, and fluid at top becomes hotter and less dense, preventing hot fluid from rising. Density difference causes force, but force is not enough to overcome viscous force.

hot

If hot fluid rises smoothly into cooler fluid, heat transfers mostly by convection, cooler fluid sinks as hot fluid rises, and hot fluid cools when it reaches top, so smooth flow goes upward in center and downward at edges.

Lorenz attractor

If hot fluid rises so quickly that it has lost little heat when it reaches cooler-fluid top, and it does not get as much heat from burner, it eventually slows flow rate {deterministic non-periodic flow}. Then system {Lorenz system} slows non-linearly. Non-linear behavior never repeats but is periodic around central points {Lorenz attractor} in multidimensional-variable space.

MATH>Computer Science>Systems>Non-linear

non-linear system

Non-linear systems {non-linear system} have terms that are multiplicative, not additive. For example, Navier-Stokes fluid-dynamics equation is non-linear. From some states, complex non-linear systems oscillate, but not perfectly, so they never repeat same state. Complex non-linear systems always have states, times, and locations from which they are unstable.

non-linear dynamics

Soliton equations have non-linear dynamics {non-linear dynamics}.

MATH>Computer Science>Systems>Substitution

substitution system

Starting with a number of cells, all cells can have one state. Cells replace cells according to neighboring states {substitution system}.

sequential substitution

Starting with a number of cells, all cells can have one state. Cells replace first cells reached by search, according to neighboring states {sequential substitution system}.

MATH>Computer Science>Systems>Substitution>Tag System

tag system

In a fixed number of cells, all cells can have one state. System removes cells at one end and adds cells at other end {tag system}, depending on cells removed.

cyclic tag system
In a fixed number of cells, all cells can have one state. System removes cells at one end and adds cells at other end {cyclic tag system}, depending on cells removed.

MATH>Computer Science>Systems>Artificial Intelligence

artificial intelligence
Programs {artificial intelligence}| (AI) can embody logical processes and have memory. Artificial-intelligence systems have sets {knowledge base} {database} of classes or representations. Reasoning methods {control strategy} select rule or representation, check rule conditions, and track used and available rules. AI has rule {operation} sets. AI has search mechanisms {graph search}, which apply rule at nodes to get to new nodes. Rules or predicates have attributes, sets, objects, events, values, or subsets.

expert system
Artificial-intelligence systems {expert system}| can mimic experts. Expert systems can store rules with which to classify. Expert systems can make decisions based on utility-function optimization. Expert systems can use networks with weighted connections. Sets of IF/THEN statements can make output from input.

inquiring system
Systems {inquiring system} have boundaries, communication methods, goals, and comparison methods, such as distinguishing or generalizing. Systems modify based on success or failure, build language, have memory, take perspective, change perspective, make models, and observe self.

MATH>Computer Science>Systems>Catastrophe Theory

catastrophe theory
Catastrophes {catastrophe theory}| are space discontinuities [Thom, 1968] [Thom, 1972] [Waddington, 1968] [Woodcock and Davis, 1978]. Discontinuity type depends on dimension number. Number of parameters determines how many system states are possible. Actual behavior depends on present state and past history.
transformations
Spaces with four or less dimensions allow seven discontinuous transformations: fold, cusp, swallowtail, butterfly, parabolic umbilic, elliptic umbilic, and hyperbolic umbilic. No other catastrophe types are possible. Discontinuities can appear in continuous-equation systems.

fold catastrophe
Folds make a discontinuity line between two planes {fold catastrophe}. It involves one dimension and only one state. From fold point, either stable or unstable behavior can happen.

cusp catastrophe
Folds along two dimensions make two discontinuity lines, which meet at a point between three planes {cusp catastrophe}. From meeting-point state, states can diverge on both folds, with no middle behavior between the states. Different positions and directions make different state changes {hysteresis, catastrophe}.

butterfly catastrophe
Folds along four dimensions make four discontinuity lines, which meet at a point between five planes {butterfly catastrophe}.

swallowtail catastrophe
Folds along three dimensions make three discontinuity lines, which meet at a point between four planes {swallowtail catastrophe}.

elliptic umbilic catastrophe

Folds along three dimensions make three discontinuity lines, which meet at a line {elliptic umbilic catastrophe}. It involves three dimensions and only two states.

hyperbolic umbilic catastrophe
Folds along three dimensions make three discontinuity lines, which meet at a line {hyperbolic umbilic catastrophe}. It involves three dimensions and only two states.

parabolic umbilic catastrophe
Folds along four dimensions make four discontinuity lines, which meet at a surface {parabolic umbilic catastrophe}. It involves four dimensions and only two states.

MATH>Computer Science>Systems>Chaos Theory

chaos theory
Even complex systems have parts with simple processes. Chaotic systems have parts with no chaos. Small initial-condition changes can cause large result changes {chaos theory}| [Lorenz, 1963].

butterfly effect
Non-linear complex systems are sensitive to initial conditions, as if butterfly flight in one place affects weather pattern in another place {butterfly effect}| [Gleick, 1987] [Prigogine and Nicolis, 1989] [Prigogine and Stengers, 1984] [Prigogine, 1980] [Ruelle and Takens, 1971] [Ulam, 1976] [Li and Yorke, 1975].

periodicity in systems
Systems can have processes that regularly repeat {periodicity}|. Increased repetition rate {period doubling} or system size can lead to chaos, as wavelength and space decrease. Systems can make matter or energy pulses {intermittency}. Systems can have periods that are not exact {quasiperiodicity}. Energy or mass dissipation cancels or removes conflicting motions and results in changes along one dimension. Periodicity is only in that dimension.

MATH>Computer Science>Systems>Complexity Theory

complexity in systems
Systems can have many and/or different states, objects, events, dependencies, and interactions {complexity, system}. Complex systems share information among objects [Goodwin, 1994] [Kauffman, 1993] [May, 1976] [Pagels, 1988] [Prigogine and Nicolis, 1989] [Prigogine and Stengers, 1984] [Prigogine, 1980] [Smale, 1967].

hierarchy
Having more parts and relations allows part and relation levels.

emergence
Having more parts and relations allows new part and relation combinations.

laws
The many and varied complex-system relations make conservation laws, constancies, covarying elements, and other regularities.

flows and circuits
The many complex-system relations propagate changes throughout system.

states
Systems can return to initial or previous state or reach terminal state, so system halts or repeats. Terminal states are typically undesirable, unstable, or otherwise break system, so system must protect against them. Physical complex systems can self-destruct. They must have mechanisms to prevent halting, repeating, and breakdown or extricate themselves from such situations. Complex systems typically are stable for only some states. Stable complex systems have resting or default states.

non-isolation
Complex systems can have complex input and output.

non-isolation: energy
Stable physical complex systems require energy sources and regulate energy input and output between system and environment.

non-isolation: environment
Physical complex systems can work in only one environment type.

complexity theory

Regularity enumeration can measure complexity {complexity theory}| [Kauffman, 1993].

computational complexity

Algorithm classes require time ranges {computational complexity theory}.

algorithmic information

System complexity measures {algorithmic information content} {algorithmic complexity, system} {Kolmogorov complexity, system} can be numbers of bits for smallest program that can run on universal Turing machines and produce same output.

algorithm

Programs produce output from input and algorithm. Theory predicts facts from data and formulas. Algorithms and formulas are similar.

number

Random numbers have programs about as long as themselves. Information has no redundancy and cannot compress {irreducible information}.

proof

Infinitely many mathematical results require algorithms or proofs as large as output and so have no useful proofs. For example, axioms have no proof. Therefore, the principle of sufficient reason is not always true.

halting problem

In general, whether programs will stop, or not, is impossible to predict {halting problem}|, as shown by Turing.

Turing machine

Turing machines must have at least one rule that leads to STOP, must not move to non-existent state, must use and make only coded sequences of marked and blank squares, and must have at least one marked square on output side. Turing machines have input and rules. Number of Turing machines and number of inputs are both infinite.

Many Turing machines never reach STOP.

If people can prove that Turing machine with some input reaches or does not reach STOP, people can make complex Turing machines that include that Turing machine and answer the question whether Turing machine stops. People can program complex Turing machines to make same mark, as long as that Turing machine does not stop. Complex Turing machine can include all Turing machines, so then all Turing machines can mark definite answers for inputs, whether they stop or not. Therefore, one algorithm decides same Turing machine reaches STOP, and one algorithm decides same Turing machine does not reach STOP. However, only one algorithm is true, and the other is false. Therefore, it is impossible to prove that Turing machines will reach STOP.

halting problem

Possible Turing machines have representations as natural numbers. Possible inputs have representations as natural numbers. If numbers of inputs and machines are equal, a square natural-number array can represent all possible Turing machines and inputs. See Figure 1.

Look at sequence, to take diagonal slash, on square-array main diagonal. See Figure 2.

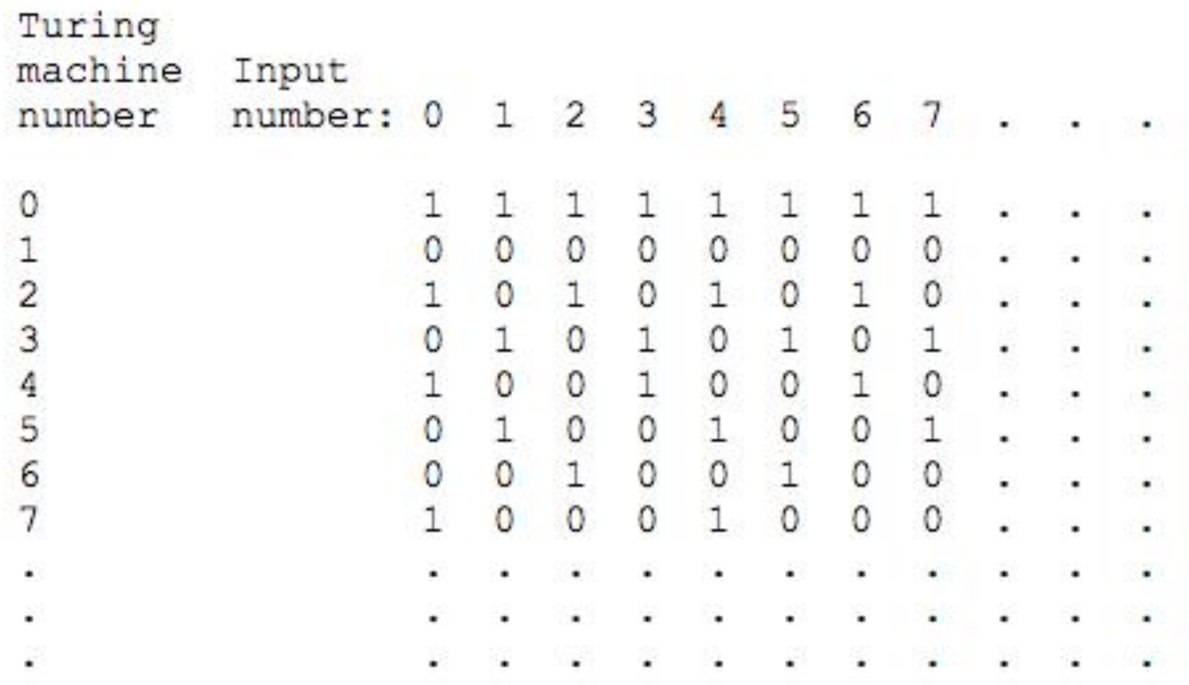
Change marks for numbers in diagonal sequence. See Figure 3.

This sequence is not the same as any row or column sequence, because it differs from first row and column at first number, differs from second row and column at second number, and so on. If halting problem is solvable, this sequence can represent possible Turing machine/input combination, because first part can be legitimate Turing machine and second part can be legitimate input. However, array must contain all possible sequences, because array has all possible Turing machines and all possible inputs. Contradiction makes halting problem not solvable in general.

omega

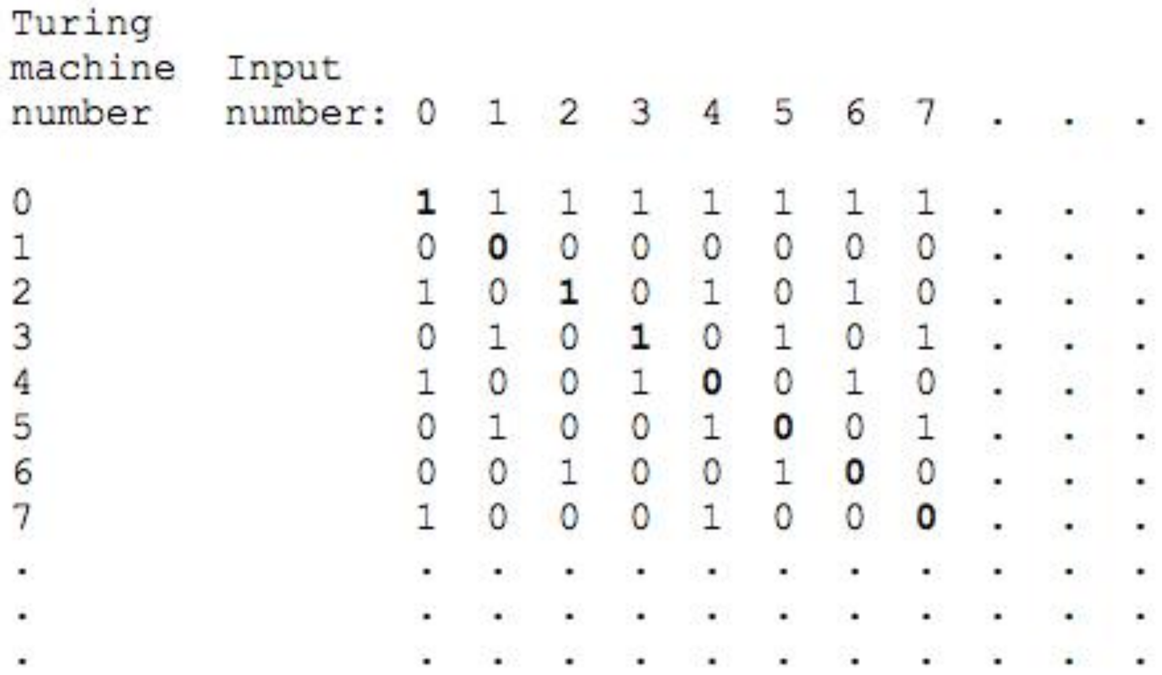
Programs have halting probabilities {omega, number} {Chaitin number}. Data bits can be input to program until program does not request another bit, because it has stopped. Random-bit input stop program after different numbers of input bits. Probability that program stops is 0.5 raised to number of bits. To find total halting probability, add random-input-experiment probabilities. Using more random-input experiments can approach omega.

Figure 1



Turing machine number	Input number:	0	1	2	3	4	5	6	7	.	.	.
0		1	1	1	1	1	1	1	1	.	.	.
1		0	0	0	0	0	0	0	0	.	.	.
2		1	0	1	0	1	0	1	0	.	.	.
3		0	1	0	1	0	1	0	1	.	.	.
4		1	0	0	1	0	0	1	0	.	.	.
5		0	1	0	0	1	0	0	1	.	.	.
6		0	0	1	0	0	1	0	0	.	.	.
7		1	0	0	0	1	0	0	0	.	.	.
.		.	.	.	.	.	.	.	.	.	.	.
.		.	.	.	.	.	.	.	.	.	.	.
.		.	.	.	.	.	.	.	.	.	.	.

Figure 2



Turing machine number	Input number:	0	1	2	3	4	5	6	7	.	.	.
0		**1**	1	1	1	1	1	1	1	.	.	.
1		0	**0**	0	0	0	0	0	0	.	.	.
2		1	0	**1**	0	1	0	1	0	.	.	.
3		0	1	0	**1**	0	1	0	1	.	.	.
4		1	0	0	1	**0**	0	1	0	.	.	.
5		0	1	0	0	1	**0**	0	1	.	.	.
6		0	0	1	0	0	1	**0**	0	.	.	.
7		1	0	0	0	1	0	0	**0**	.	.	.
.		.	.	.	.	.	.	.	.	.	.	.
.		.	.	.	.	.	.	.	.	.	.	.
.		.	.	.	.	.	.	.	.	.	.	.

Figure 3

```
from Figure 2 diagonal
1  0  1  1  0  0  0  0  .  .  .

to new sequence by changing each position
0  1  0  0  1  1  1  1  .  .  .
```

logical depth

Process or system complexity can be number {logical depth}| {depth of argument} of steps from original input to final output [Bennett, 1988] [Herken, 1988] [Norretranders, 1998] [Goodwin, 1994] [Kauffman, 1993] [Koch and Laurent, 1999] [May, 1976] [Pagels, 1988] [Prigogine and Nicolis, 1989] [Prigogine and Stengers, 1984] [Prigogine, 1980] [Smale, 1967]. Steps can be logical steps from premises to conclusions, cause-and-effect relationships, or algorithm steps.

strange attractor

Phase-space points {strange attractor}| can be, not equilibria or periodic loops, but infinitely long lines in confined regions. Strange attractors are stable, can have few dimensions, and are periodic but not exactly periodic.

MATH>Computer Science>Systems>Computer Vision

computer vision

Stationary or moving cameras capture images, and associated hardware and software perform image analysis to find features, align images, extract descriptors and objects, and classify objects and motions {computer vision}.

Euclidean distance

Two points, in image or in physical or mathematical space, or two corresponding points, on two simultaneous images from two stereo cameras or on successive images from same camera, have distances {Euclidean distance} between them.

monogenic signal

Signals {monogenic signal} can have phase, distance, time, orientation, or amplitude information.

visibility problem

Image object parts typically hide other object parts, so objects have visible parts and hidden parts {visibility problem}. Object-recognition algorithms recognize objects using parts. Image-generation algorithms show nearest object parts.

voxel

Three-dimensional space has volume pixels {voxel}. Voxels can be opaque, translucent, or transparent. Voxel transparency level can be a parameter {integral alpha}. Perceived pixel color P depends on integral alpha, voxel color intensity A, and background pixel color intensity B: P = alpha * A + (1 - alpha) * B, where A and B are red, green, or blue. Integral alpha is the same for all three color channels. For transparent voxels, alpha is zero, perceived color is background color, and voxel has no effect. For opaque voxels, alpha is one, perceived color is voxel color, and background has no effect. Therefore, systems can multiply alpha and voxel and background colors {premultiplied alpha} before compositing, for efficiency.

Volumetric displays can have multiple parallel display planes or a rotating plane to sweep out volume.

MATH>Computer Science>Systems>Computer Vision>Algorithms

computer vision algorithms

Computer vision uses image-capture-and-processing algorithms {computer vision algorithms} to find features and objects.

image capture

Cameras capture images pixel by pixel, with white, red, green, or blue intensity. Image algorithms can rotate, change dimensions, enhance brightness, and enhance contrast.

counting pixels

Counts light or dark pixels.

thresholding

Converts gray to black or white.

segmenting

Locates or counts textures and/or parts.

detecting edges

Finds object edges.

discovering blobs

Inspects image for connected white or black pixels to establish image reference points and regions.

recognizing components

Extracts simple three-dimensional geometric figures from image.

recognizing patterns

Locates features or objects, which may have different rotations or sizes and overlap or occlude each other.

MATH>Computer Science>Systems>Computer Vision>Algorithms>Clustering

clustering algorithms

Image points belonging to same feature are near each other. Parameter-space points belonging to same or similar features are near each other. Nearest neighbors can form point, line, or angle clusters {clustering algorithm}. Nearest-neighbor algorithms measure Euclidean distance, or other distance metrics, to find nearest-neighbor points and make clusters. Using distance measure can group features.

types

Clustering algorithms include hierarchical, self-organizing maps, K-means, fuzzy C-means, and expectation maximization. Error-weighted clustering retrofits clustering algorithms to use error-propagation information.

Self-organizing maps can group into equal categories.

process

Clustering can start with all features in one category and split features off {divisive clustering} or start with individual features and group them into classes {agglomerative clustering}. Clustering can use feature information to start or modify clustering {supervised clustering} or use only distances {unsupervised clustering}.

distance metric

Different hierarchical clustering methods use different ways to calculate distance between two clusters.

Minimum distance between cluster features {nearest-neighbor clustering} {single-linkage clustering} makes loose clusters with long branches containing few samples. Maximum distance between cluster features {furthest-neighbor clustering} {complete-linkage clustering} makes tight similar-size clusters with short branches.

Clustering can use average or median distance between features {unweighted pair-group method average} (UPGMA) or weighted-average point {centroid, cluster}.

For widely varying cluster sizes, average distance between features has weight {weighted pair-group average}: cluster feature number.

Clustering can use distance between cluster averages {within-groups clustering}.

distance metric: city-block distance

Distance measure between structure-space points is the same as Manhattan distance.

distance metric: Lp-metric

Distance measure between structure-space points is the same as Minkowski distance.

distance metric: Mahalanobis distance

Structure-space points have distances between them.

distance metric: Manhattan distance

Distance measure between structure-space points is the same as city-block distance.

distance metric: Minkowski distance

Distance measure between structure-space points is the same as Lp-metric.

supervised method

Classification can use already known patterns and clusters.

hierarchical clustering

Unsupervised, agglomerative clustering method {hierarchical clustering} measures distances between all points and places features in cluster hierarchy that looks like tree diagram {dendrogram, clustering}. Hierarchical clustering can make hierarchies and assign shared feature. Single features, and feature groups, are clusters. After calculating distance for cluster pairs, the closest clusters merge into one cluster, reducing cluster number by one. After calculating distance again for cluster pairs, the closest clusters merge into one cluster, and so on, until all features are in one, top-level cluster. Hierarchical clustering methods include centroid linkage, complete linkage, single linkage, and Ward's method. Ward's method calculates cluster mean and sum of squared differences from mean, for all cluster points. The next cluster is pair that gives smallest increase in sum of squared differences.

non-hierarchical clustering

Features can also be in different classes with no hierarchy {non-hierarchical clustering}.

non-hierarchical clustering: k-means clustering

Non-hierarchical, unsupervised method places features into a fixed number of clusters. First, it randomly assigns features to clusters. It calculates distances between feature pairs in cluster to find average cluster expression vector. It calculates distances between cluster pairs using average cluster expression vector. Features move to other clusters in turn, and method calculates all feature distances. Features stay in new cluster if distances between cluster feature pairs and between cluster pairs decrease.

Supervised k-means clustering first assigns features to clusters based on feature information and then proceeds as for unsupervised k-means clustering.

non-hierarchical clustering: self-organizing map

Non-hierarchical, unsupervised method places features in cluster based on nearness to cluster reference vector. First, cluster number is set. Starting from random vector, cluster reference vector converges by better partitioning feature data. It calculates distances between each expression vector and reference vectors and assigns feature to one cluster.

non-hierarchical clustering: principal component analysis

Non-hierarchical, unsupervised methods {principal component analysis, vision} (PCA) {singular value decomposition, vision} can combine feature dimensions linearly to reduce expression-space dimensions, remove redundancy, and average data. Similar unsupervised linear methods {factor analysis, computer} (FA) can look for significant factors in factor sets and find one, two, or three combined factors. It includes principal component analysis, correspondence analysis, spectral mapping, and non-linear mapping. Another similar technique {principal coordinate analysis} combines coordinates to make fewer dimensions. PCA, factor analysis, and principal coordinate analysis project data onto lower-dimension space by eliminating dimensions and dimension combinations with low significance. Number of remaining dimensions gives number of clusters to use for k-means clustering or self-organizing maps.

non-hierarchical clustering: support vector machine

Supervised clustering methods can use feature-vector training sets, group similar-function features into clusters, and group other-function features outside cluster. It tries to find cluster-specific features. Test features are in or outside cluster.

Features split space into two regions by making boundaries {hyperplane}. Hyperplanes can partition features so regions {soft margin} near hyperplanes have ambiguous examples.

Features can partition higher spaces {feature space}, mapped from feature vectors. Feature-space distances are metric and use special function {kernel function}. The best partition typically increases kernel function from simple to complex.

class analogy

Class analogy is a SIMCA method.

cluster sampling

Sampling can be from clusters equally.

cluster significance analysis

Using discrete or continuous data and embedded data can put compounds into groups by activity level. CSA locates small clusters in large spaces.

Cone and Hodgkin similarity index

Method measures molecular similarity.

discriminant-regression model

Model locates small clusters in large spaces.

distance-b program

Method locates small clusters in large spaces.

Jarvis-Patrick method

Structures can cluster in large databases by rating different features by similarity.

k-nearest neighbor

Supervised method calculates new object distance from all other objects to locate small clusters in large spaces.

partitioning

Process merges individuals into groups or splits whole into clusters.

similarity measure

Value can compare distances.

single-class discrimination

Method locates small clusters in large spaces.

Soft Independent Modeling of Class Analogies (SIMCA)

Supervised method uses model to find region boundaries or envelopes and locates small clusters in large spaces.

trend-vector analysis

Using activity and descriptor correlation vector ranks different features by similarity.

Haar wavelet
Image regions have sub-regions, which have different numbers of same-intensity pixels. Matrices can represent regions, and matrix cells can represent sub-regions. Cell values are number of same-intensity pixels. Matrices then have differences between cell sub-region values {Haar-like feature}, and matrices represent wavelets {Haar wavelet}. Regions are clusters or neighborhoods, such as rectangles or spheres, and so each region type has unique wavelet. Region sets have waves.

outlier algorithms
Images have points that do not cluster and do not belong to features. Outlier algorithms {outlier algorithms} use linear regression and best-fit to find outliers and discard them.

Potts model
Methods {Potts model} can minimize energy by making two states either equal or unequal, with constant penalty for unequal and no penalty for equal.

MATH>Computer Science>Systems>Computer Vision>Algorithms>Depth

belief propagation
Message-passing labeling inference algorithms {belief propagation} can compute many graphical-distribution statistics, each with few values, to calculate disparities by finding large-neighborhood minima. Belief propagation uses sum-product to find minimum or max-product to find maximum a posteriori (MAP) estimate. For stereo vision, Markov random-field model describes disparity, and inference algorithm determines nodes.

binary space partitioning
Algorithms {binary space partitioning} (BSP) can recursively divide space or polygon into two regions using hyperplanes, to make halves, quarters, eights, sixteenths, and so on.
orientation
Hyperplanes can have any orientation, making unequal regions. Hyperplanes that cut at medians make both regions equal.
polygons
Polygons can have angles greater than 180 degrees {reflex angle, polygon} or less than 180 degrees. Dividing polygon recursively makes regions with angles less than 180 degrees {convex set}.
recursion
Recursion steps define trees {BSP tree} and make stored lists {visibility list} that order polygons from front to rear. Convex sets become smaller until they include only point {BSP-tree leaves}.
multiple hyperplanes
Binary space partitioning can use hyperplane pairs or triples for cuts. Hyperplane pair divides space into four regions {quadtree}. Hyperplane triple divides space into eight regions {octree}.

iterative closest points method
Methods {iterative closest points method} can use point-sample clouds to align two images.

k-dimensional tree
Modified BSP-tree algorithms {kd-tree} {k-dimensional tree} can use only hyperplanes perpendicular to space axes and use axis sequences, typically splitting at axis or polygon medians. If space has k axes, kd-tree has k cuts and tree branchings. kd-tree algorithms are better for searches using nearest neighbors, because they match space coordinate parameters and split hyperplanes go through points. Because hyperplanes are across axes, all regions have nodes or points.

least-squares adjustment
Gauss-Helmert methods can use least squares {least-squares adjustment} to estimate best fit.

painter's algorithm

Painters paint background first, then layer objects on canvas from rear to front {painter's algorithm}, drawing over things that become behind.

rectification of image
Stereo images project onto aligned image plane by transformation {rectification of image}.

self-calibration
Cameras can use epipolar transform and absolute conic image in Kruppa equation to find standard metric {self-calibration}.

self-localization
Robots can find their locations in environments {self-localization}, using self-localization alignment methods (SLAM).

z-buffering
Algorithms {z-buffering} {depth buffering} can store object-part depths for image generation or object recognition. z-buffers {depth buffer} represent two-dimensional images, for object identification. z-culling algorithms compare object-part depths and store object with smallest depth in buffer. The same visual angle covers more space farther away. To control for increasing spread with increasing distance, use variant w-buffers.

MATH>Computer Science>Systems>Computer Vision>Algorithms>Features

feature detection methods
Feature detection can use point matching based on feature or boundary matching {feature detection methods}: corner detection, scale-invariant features (SIFT), and speeded-up robust features (SURF). Features are good image-category descriptors.

feature detection algorithms
Algorithms {feature detection algorithms} can detect features.
descriptor
Properties {X-variable, vision} {X descriptor, vision} can describe features and have components.
canonical factor analysis
Factor analysis has a basic method.
centroid method
Factor analysis can use centroids.
Correlation Analysis
Properties and features have relationships.
correspondence factor analysis
Factor-analysis methods can use variable frequencies relative to properties, find chi-square values, and find principal components.
disjoint principal component
Principal components can be independent.
eigenvalue-one criterion
Thresholds can be how many components have eigenvalues greater than one.
eigenvector projection
Unsupervised linear methods can find factors.
Evolutionary Programming
Models can add and subtract randomly selected variables, with crossing-over, and evaluate for "fitness" or best fit. Extinction can happen more or less frequently or in more or fewer species. More-frequent extinctions have fewer species. Values follow power laws, because events can cause few or many extinctions.
evolving factor analysis
Methods can analyze ordered data.
explained variance percentage
Methods can indicate component number required to reach 90% of total variance.
factorial design
Designs can try to ensure design-space sampling, even if one position varies.

Genetic Function Algorithm

Linear property sets can have different values, change values by crossing-over between related genes, and have random changes, to select best fit.

latent variable

Variables can be linear descriptor combinations.

linear discriminant analysis

Supervised methods, in which boundary surface minimizes region variance and maximizes between-region variance, can put compounds into groups by activity level.

linear learning machine

Supervised methods can divide n-dimensional space into regions using discriminant functions.

maximum-likelihood method

Factor-analysis methods can find factors.

multidimensional scaling

Metric or non-metric methods can analyze similarity or dissimilarity matrices to find dimension number and place objects in proper relative positions.

multivariate adaptive regression spline

Non-parametric methods can find factors.

Mutation and Selection Uncover Models

Models can add and subtract randomly selected variables, with no crossing-over, and evaluate for "fitness" or best fit. Low mutation rates allow natural selection to operate on populations to move toward fitter genotypes. Intermediate mutation rates cause population to move toward and away from fitter genotypes. High mutation rates make many genotypes with direction, so high mutation blocks selection processes.

For any mutation rate, if gene number is too great, change rate is too great, and organism becomes extinct {error catastrophe, extinction}. Therefore, gene number has a limit if organisms do not make new species or find new environments.

Perhaps, cells and ecosystems also have upper limits to complexity. Complexity can increase with migration or speciation.

non-linear iterative partial least squares

Unsupervised linear methods can represent data as product of score matrix, for original observations, and loading-matrix transform, for original factors.

non-linear mapping

Topological mapping factor-analysis method uses linear variable combinations to make two or three new variables.

predictive computational model

Property information can predict behavior.

principal-component analysis

Variable principal components can be linear-descriptor combinations. Unsupervised linear methods can represent data as product of score matrix, for original observations, and loading-matrix transform, for original factors. PCA factor-analysis method uses linear variable combinations to make two or three new variables. PCA reduces unimportant variables.

principal-component regression

Singular-value decomposition can find singular-value sets to predict and project regression to latent structures.

principal factor analysis

Modified PCA can find principal factors.

Procrustes analysis

Methods can identify similarity descriptor sets.

QR algorithm

Methods can diagonalize matrices.

rank annihilation

Unsupervised linear methods can find factors.

response-surface method

Three-level designs can have three factors that quantify response and factor relationships. RSM includes MLR, OLS, PCR, and PLS linear designs, non-linear regression analysis, and non-parametric methods such as ACE, NPLS, and MARS.

Scree-plot

Residual variance approaches constancy, and plotted slope levels off, depending on number of components {Scree-test, vision}.

singular-value decomposition
In unsupervised linear methods, correlation matrix can become product of score, eigenvalues, and loading matrices, with diagonalization using QR algorithm.
spectral-mapping analysis
Factor-analysis methods can first take data logarithms to eliminate outliers and then subtract means from rows and columns, to leave only variation, showing which variables are important and how much.
structure space
Spaces can have two or three principal components.
target-transformation factor analysis
Methods can rotate features to match known patterns, such as hypotheses or signatures.
Unsupervised Method
Factors and response variables can relate, without using factor information or predetermined models.

eight point algorithm
Methods {eight point algorithm} can find structures from motions.

facial expression recognition
People can recognize six basic facial expressions {facial expression recognition}: anger, disgust, fear, happiness, sadness, and surprise. Expressions have unique muscle activities {Facial Action Coding System}, grouped into Action Parts. Methods detect faces, extract features, and classify expressions. Classifying can use Gabor filters, Bézier volumes, Bayes and Bayesian network classifiers, and Hidden Markov Models.

Kalman filter
Gaussian filtering {Kalman filter} can use mean and variance parameters for normal distributions and can increase feature or pixel gain. Kalman filters are parametric, as opposed to particle filters. Kalman filters predict output from input.

local image analysis
First computer-vision stage is to find features, including invariants {local image analysis}. Invariants can be angles, local phase, and orientation.

particle filtering
Distributions can have representations as finite numbers of samples {particle, sample} defined on Markov chains {particle filtering}, rather than using parameters.

Sobel edge operator
Operators {Sobel edge operator} can detect edges.

support vector machine
Methods {support-vector machine} can detect shapes from image segmentation, using color, shape, and distances.

MATH>Computer Science>Systems>Computer Vision>Algorithms>Hill Climb

hill-climbing methods
Image features make maxima in parameter spaces. Algorithms {hill-climbing methods} can find local maxima that exceed threshold amount. If maximum is at feature parameter-space location and exceeds threshold, algorithm states that feature is in image and identifies location. Hill-climbing methods can become stuck at local maxima and fail to find more-important maxima.

Broyden-Fletcher-Goldfarb-Shanno method
Hill-climbing algorithms {Broyden-Fletcher-Goldfarb-Shanno method} (BFGS method) can improve quasi-Newton method.

Newton optimization method

Hill-climbing algorithms {Newton's optimization method} {Newton optimization method} can solve unconstrained non-linear optimization problems using function second-order partial-derivative Hessian square matrices, to find local maxima at locations where derivatives become zero.

quasi-Newton method

Hill-climbing algorithms {quasi-Newton method} can simplify Newton's optimization method by simplifying Hessian matrix.

watershed algorithm

Hill-climbing algorithms {watershed algorithm} can find maximum gradient from center pixel to eight surrounding pixels and move to pixel with maximum gradient. If new pixel is minimum or is lower than threshold, stop and assign original pixel to same group as second pixel. If new pixel is not minimal and is not lower than threshold, find maximum gradient from second pixel to eight surrounding pixels and move to pixel with maximum gradient.

MATH>Computer Science>Systems>Computer Vision>Algorithms>Imaging

image processing algorithms

Algorithms {image processing algorithms} can determine background, cluster, check error and noise, find fiducials, normalize, check crosstalk, find signals, find features, aggregate replicates, and perform statistical analysis. Statistical cluster analysis can identify classes and assign shared feature. Hierarchical clustering can cluster into hierarchies. Self-organizing maps can group into equal categories.

background determination

Determining background {background determination algorithm} involves surface-fitting algorithm that detects global background changes across array. Background subtracts from features. Algorithm finds variance within feature integration aperture after global correction. Using average feature density can overestimate background. Putting space between feature blocks can find background.

error model algorithm

Algorithms {error model algorithm} can check feature-signal noise, measure background variance, find cross-hybridization, check for spatial crosstalk and spectral crosstalk, measure normalization variation, and study replicates. Error model weights optimize signal-to-noise ratio. It has confidence value. Error flags label too-high variance, hybridization-control variance, high background, rejected pixels, bright neighbors, too-low signal-to-noise ratio, and saturated pixels.

fiducial-finding algorithm

Fiducials {fiducial-finding algorithm} are marks or shapes every 300 pixels, for automatic spot finding and row and column counting, without using quantitation-control scheme. Fiducials must be easy to distinguish from other image features and artifacts. Spotting and shrinking cause non-linear distortions and determine fiducial frequency needed. Row and column drift must be less than half the distance, five or six pixels, between features.

normalization of results

To account for labeling-amount, dye, fluorescent-detection, spotting, RNA-concentration, and sample-quantity differences, systems modify intensities {normalization, results}. Normalization allows comparison among slides and cell extracts.

types

Normalization can normalize on total intensity. Normalization can normalize on means and use ratio statistics. Normalization can use linear regression. Non-linear regression includes local regression, such as Locally Weighted Scatterplot Smoothing (LOWESS). Normalization algorithms use dilution-series controls, dye selection, filter selection, dye-quenching non-linearities, multiple gain settings, photobleaching amounts, and array-to-array normalization.

spatial crosstalk

Strong signals at features can spread to neighboring features and skew weak signals. Features must be far enough apart to prevent more than 0.1% spatial crosstalk. Algorithms {spatial crosstalk mitigation algorithm} can reduce number of strong signals adjacent to weak signals. Algorithms can weight intensity sum across feature depending on neighboring features. Algorithms can use deconvolution with particular instrument.

signal integration

Algorithms {signal integration algorithm} {signal quantitation algorithm} can use median pixel, integrate intensity, or use pixel-ratio median to generate values. First, algorithm aligns channels, if necessary. Fixed aperture is in both channels. Simultaneous scanning in both channels reduces crosstalk and allows pixel-to-pixel regression, which is more robust to defects. Integration can be only for high signal-to-noise pixel subsets.

spot-finding algorithm

Finding sites {spot-finding algorithm} can use three methods: find isolated peaks, align grid, or align centroid. To find isolated features or peaks, first shrink image to convolve it with feature model and then find isolated peaks. To align grid, first find fiducials and then fit rows and columns. To align centroid, first move off grid and find intensity-weighted centroid, if signal-to-noise feature ratio allows.

replicate aggregation

Algorithms {probe aggregation algorithm} {replicate aggregation algorithm} can average replicates.

MATH>Computer Science>Systems>Computer Vision>Algorithms>Prediction

prediction algorithms

Algorithms {prediction algorithms} can predict.

predictors

Factors, properties, or structures contribute to response values.

variable influence on prediction

Methods {variable influence on the prediction} can determine variable importance.

principal property

Variables {principal property} can be linear descriptor combinations.

non-parametric algorithms

Non-parametric algorithms can have alternating conditional expectations. Non-parametric methods {non-linear partial least-squares, vision} can find least squares.

outlier algorithms

Normal-distribution-outlier tests {Dixon's Q-test, vision} can measure ratio of smallest and largest differences.

Normal-distribution outlier tests {Grubbs' s-test, vision} can compare absolute values, of differences between mean and value, divided by standard deviation, to T distribution value.

network

Kohonen topology-preserving network mappings can retain topology. Topological indexes can represent graphs as numbers. Topological Tanimoto indexes can represent graphs as numbers.

rule induction system

IF/THEN statements {rule induction system, vision} can make output from input.

MATH>Computer Science>Systems>Computer Vision>Algorithms>Regression

regression algorithms

Factors, properties, or structures can correlate with response values {regression algorithms}.

regression

Regression analysis finds property and structure relationships. Multiple linear regression measures linear-component dependence on properties and finds descriptor coefficients. Non-linear regression is a parametric method that finds descriptor coefficients. Ridge regression is another regression method.

correlation

Factors can correlate, with correlation coefficients. Variance-covariance matrices {correlation matrix, vision} are complete, symmetric, square matrices that use property values and structure values, which can scale to normalize data. Partial least-squares can simplify variance-covariance matrix {matrix diagonalization, vision} {matrix bidiagonalization method, regression}.

Spearman rank correlation coefficient can measure molecular similarity.

least-squares

Ordinary least-squares {classical least-squares, vision} {least-squares regression, vision} {linear least-squares regression, vision} {multiple least-squares regression, vision} {multivariate least-squares regression, vision} can find

descriptor coefficients by minimizing distances between values and regression line. Inverse least-squares inverts fitting method.
adaptive least-squares
Adaptive least-squares modifies ordinary least-squares by weighting or classes.
adaptive least-squares: fuzzy
Features can be in different classes with different weights.
partial least-squares
PLS uses least-squares to find independent variables and dependencies among variables. PLS maximizes latent-variable and observable covariation. PLS diagonalizes variance-covariance matrix. Multi-block PLS uses groups. Kernel algorithm is about covariation.
partial least-squares: Comparative Molecular Field Analysis
Partial least-squares methods (CoMFA) can analyze grids around sites and find grid-point interactions, to make sampled-point descriptors.
partial least-squares: Generating Optimal Linear PLS Estimations
GOLPE uses PLS and D-optimal design to select variables and then cross-validates.
partial least-squares: SAMPLS algorithm
Partial least-squares and trend vector analysis can work together.
non-least-squares
Non-least-squares methods can detect non-linear relationships.

MATH>Computer Science>Systems>Computer Vision>Algorithms>Statistics

statistical algorithms
Statistics has algorithms {statistical algorithms}.
best linear unbiased estimation
Estimates can give smallest variances among estimators.
mean square error
SSE / (observation number + factor number - 1).
SSE
Errors or residuals cause sum of squares of differences between observed and predicted responses.
SSR
Regression causes sum of squares of differences between observed and mean.
SST
Sum of squares of differences between predicted and mean makes total: SST = SSE + SSR.
standard error
Standard error is MSE square root.

MATH>Computer Science>Systems>Computer Vision>Algorithms>Transform

Hilbert transform
Transforms {Hilbert transform} can be for one-dimensional signals.

Hough transform
Features have parameters, such as length, angle, color, distance, radius, and angle. Parameters have continuous-value ranges, such as lengths from millimeters to meters, or discrete-value sets, such as color categories. Features have parameter values that make vectors. For example, if parameter number is four, features have four-vectors, such as (0, 0, 0, 0), (3, 0, 0, 0), or (4, 2, 1, 3).
space
Hough spaces have dimension number equal to parameter number. In the example, Hough space has four coordinates. Hough space points can represent features.
feature extraction
Objects whose feature vector lies near feature point have that feature.
feature extraction: accumulator
Feature extraction can use voting {Hough transform}, to accumulate weights at feature points in Hough space {accumulator space} (Paul Hough) [1962]. After voting, if weight passes threshold at feature point, image has feature.
lines

Parameterized standard line, circle, or ellipse test sets establish feature-point coordinates in accumulator space. For lines, accumulator space can use polar-coordinate radius and angle. Edge-detector algorithms can pre-process images to find edges. Hough transforms can group edge points into lines, circles, or ellipses for identification (Richard Duda and Peter Hart) [1972] (Dana H. Ballard) [1981].

pyramid transform

Transforms {pyramid transform} can be for three-dimensional signals and takes high-resolution images and makes low-resolution images.

Radon transform

One-dimensional Hilbert transforms {Radon transform}, at specific orientations, can transform multi-dimensional signals.

Riesz transform

Transforms {Riesz transform} can be for two-dimensional signals.

MATH>Computer Science>Systems>Computer Vision>Algorithms>Validation

validation algorithms

Methods {validation algorithms} can check correlations, predictions, and designs.

bootstrapping validation

Method can use only internal data.

external validation

Other data can pair with model to predict activity.

cross-validation

For all data subsets, algorithm {leave-one-out, vision} {leave-groups-out, vision} can remove data subset and calculate remainder, such as for drug validation.

cross-validation: correlation coefficient

Method can validate and predict data.

fitness function

Validation method {lack-of-fit method} {jackknife validation method, vision} can measure fit, such as for drug validation.

Fisher F-test

Validation method can use F test.

other methods

Methods can be for drug validation {predictive residual sum of squares, vision} {scrambling dependent Y-values, vision} {standard deviation method, vision} {standard error of predictions, vision} {standard error of regression, vision}.

MATH>Computer Science>Systems>Computer Vision>Brain Theory

brain theory

Brain theories {neural modeling} {brain theory} {computational neuroscience} model membrane currents, chemical changes, network oscillations, microcolumns, macrocolumns, and cell configurations to study learning and memory (Eric L. Schwartz).

viewpoint dependence

Real vision stores different viewpoints and matches similar schema.

single-neuron models

Neuron membranes have fast sodium-ion out-currents and later slower potassium-ion in-currents, as well as calcium ion, chloride ion, and other chemical flows, which affect action potential, adaptation, and shunting.

Dendrites and axons have structures and patterns.

Synapses have ion and chemical flows.

synapse plasticity

Synapse structure and physiology change over time with electrical and chemical flows. Feedback can alter weights in Hebbian learning.

More stable synapses learn and forget slower. Less stable synapses learn and forget faster. Systems use slow and fast plasticity combinations.

neural coding

Neurons have preferred stimuli. Neural coding can use instantaneous or average impulse frequency for rate coding. Impulses can code intensity, which can represent stimulus amplitude or stimulus probability.

Neuron signaling uses minimal number of impulses to convey information (Horace Barlow).

neural inhibition and excitation

Inhibition can subtract or divide. Excitation can add or multiply. Adding and subtracting accumulate same stimulus type, to pass or not pass threshold and determine whether to perform action. Multiplying and dividing represent stimulus interactions and feature pairing, to allow object detection or recognition.

neuron development

Neurons, axons, and dendrites migrate and grow. Migration and growth use hormone and growth-factor chemical gradients. For efficiency, wiring patterns are optimal in spacing and number {minimal wiring hypothesis}.

Sense physiology uses Bayesian inference, to reflect conditional rules.

neural networks

Neurons connect specifically to each other and use recurrence. Models use neuron pairs.

memory

Memory can be associative or content-addressable. Hippocampus models are for long-term memory. Prefrontal-cortex models are for working memory. Memory can use phase synchrony and wave resonance.

cable theory

Electrical cables have resistance and capacitance, which determine oscillation time and dissipation length. Partial differential equations (William Thompson) are similar to wire heat-conductance equations (Fourier). Dendrites, cell bodies, and axons are like cables and have capacitances and resistances in parallel and series {cable theory} (Wilfrid Rall). Fibers have resistance because cytoplasm and membrane are not good conductors. Fibers have capacitance because membrane phospholipid bilayer does not conduct but has charge polarization.

Gabor function

Functions {Gabor function}, derived from neuron response frequencies, can represent neuron location, width, length, orientation, and frequency as wavelet. Gabor functions represent neuron types. Neural nets or systems are Gabor-function configurations.

Hopfield net

Units representing neurons can have binary outputs off or on, use input thresholds, and be in networks. Neural networks {Hopfield net} (John Hopfield) can use recurrence to iteratively determine final values.

convergence

Outcomes are locations and have values. Hopfield nets converge on local minima among set.

training

Training with images can determine locations that represent standard features or objects.

recognition

If local minimum matches location representing feature, Hopfield net recognizes feature in images. In particular, input that is only feature index or cue can lead to the feature, so Hopfield nets can act like memory system {content-addressable memory, Hopfield net}.

Ising model

Statistical-mechanics models {Ising model} can use pairs of +1 or -1 spins (Ernst Ising) [1925]. Pair spins can have same or different alignment. Pair energy is product of spins: +1 * +1 = 1 = -1 * -1, or +1 * -1 = -1 = -1 * +1.

Network can show spin interactions. Nodes are particles with spins, and connection edges are interaction energies. Regions, and whole systems, have total energy and are system states.

Neural nets have binary units, +1 and -1, as nodes and have connections among neurons. Total energy is system state.

MATH>Computer Science>Systems>Computer Vision>Geometry

conformal geometry

Geometry represents Euclidean space as n-dimensional sphere {conformal geometry}, not as vector space. Operations are linear. It has projective geometry.

epipolar constraint
Stereo-vision constraints {epipolar constraint} can reduce searching to along image curve {epipolar line}, using fundamental matrix, which gives camera relative orientation and position, or essential matrix, which describes epipolar geometry using focal length, chip size, or optical-center coordinates. Other camera has focal-point projection {epipole}. First-camera pixels have corresponding pixels on second-camera epipolar line.

homography of image
Images have vertical orthogonal planes {homography} at focal points.

MATH>Computer Science>Systems>Computer Vision>Geometry>Depth

disparity of images
The same scene point is at different pixel coordinates {disparity of images} in rectified images from two cameras, with distance between them. Disparity is directly proportional to depth.

disparity rate
Two images have disparities between corresponding pixels and have disparity-change rates {disparity rate}.

MATH>Computer Science>Systems>Computer Vision>Geometry>Features

image features
Image objects have special points, lines, or angles {image features}, whose enumeration or configuration can describe objects. Training on standard images can teach object features and configuration. Features have parameters, such as length, angle, color, and distance. Feature scale, noise, illumination, and distortion can differ.

augmented reality
Camera image can have virtual shapes superimposed on it {augmented reality}, to serve as landmarks or features.

curve functions
Curves have models {curve functions}. Hyperbolas have curvature, arc length, and separation. Clothoid curves have arc length related to bending. G2-splines have arc length related to bending. B-splines are closed curves. Fifth-degree Cartesian polynomials are closed curves. Polar splines are closed curves.
Three-dimensional curves use ellipsoids, spheroids, cylinders, quartics, and splines and try to find optimum position, orientation, scale, and mathematical function.
Principal component analysis can find axes.

descriptor
Image features {descriptor, image}, such as points and regions, relate to object recognition. Features can be independent or combine. Descriptors include binary, spatial, shape, texture, and local colors. Texture can be gradient location-orientation histogram. Corner detection, SIFT, and SURF use descriptors.

geon
Three-dimensional spheres, cubes, cylinders, cones, and wedges {geon} can be object-representation components. Different geon types look different from different viewpoints {viewpoint-invariance, geon}. Occluded, overlapped, noisy, blurry, or deformed geon types look different from other occluded, overlapped, noisy, blurry, or deformed geon types {stability, geon}. Object representations have geons and relations among geons {recognition-by-components, geon} (Irving Biederman) [1987].

Monge patch
Small areas {Monge patch} can have three-dimensional image patterns.

MATH>Computer Science>Systems>Computer Vision>Tasks

computer vision tasks

Computer vision tasks {computer vision tasks} include content-based image retrieval, depth perception, egomotion, event detection, identification, image restoration, indexing, learning, object recognition, pose estimation, scene reconstruction, and tracking.

art gallery problem

Find minimum-length path through art gallery from which one can see all pictures. Watchman-route problem, safari problem, zookeeper problem, touring-polygons problem, and parts-cutting problem have same type. Rubberband algorithms solve them.

barcode reading

Decode 1D and 2D codes.

character recognition

Recognize serial numbers, words, and phrases.

content-based image retrieval

In image sets, find images with content, such as text, number, object, or image.

dense stereo vision

Two cameras, with known or unknown separation and angle, can find scene-point depths.

depth perception

One eye can use linear perspective, motion parallax, interposition, shading, relative size, relative height, aerial perspective, texture, and three-dimensional-structure motion. Two eyes can use convergence and binocular disparity.

egomotion

Calculate camera three-dimensional motion.

event detection

Find abnormal or special feature or property in images.

gauging

Measure object dimensions.

human-machine interface

Algorithms allow human and robot interaction.

identification

Match individual image to stored image.

image restoration

Remove noise using low-pass filters, median filters, or image models.

kinematic chain

Rigid bars connect by sliding or rotating joints.

object recognition

Recognize learned object type at image location.

optical flow

When camera or person moves, scene flows past. Lucas-Kanade method, Horn-Schunck method, Nagel-Enkelmann method, correlation method, and block-matching method use variational methods to find optical flow.

pose estimation

Find object position or orientation.

scene reconstruction

Using several scene images, calculate three-dimensional model.

structure from motion

Motions cause disparities and disparity rates, which can reveal structure. Bundle-adjustment algorithms can find three-dimensional scene structure and camera trajectories.

First, projective reconstruction can construct projected structure, then Euclidean upgrading can find actual shape. Affine reconstruction uses Tomasi-Kanade factorization.

template matching

Find, match, and/or count patterns.

tracking

Follow object velocities and directions.

art gallery problems

Find minimum-length path through art gallery from which one can see all pictures {art gallery problems}. Watchman-route problem, safari problem, zookeeper problem, touring-polygons problem, and parts-cutting problem have same type. Rubberband algorithms solve them.

dense stereo vision
Two cameras, with known or unknown separation and angle, can find scene-point depths {dense stereo vision}.

egomotion
Calculate camera three-dimensional motion {egomotion}.

human-machine interface
Algorithms {human-machine interface} allow human and robot interaction.

image restoration
Remove noise using low-pass filters, median filters, or image models {image restoration}.

kinematic chain
Rigid bars connect by sliding or rotating joints {kinematic chain}.

optical flow
When camera or person moves, scene flows past {optical flow, camera}. Lucas-Kanade method, Horn-Schunck method, Nagel-Enkelmann method, correlation method, and block-matching method use variational method to find optical flow.

pose estimation
Find object positions and orientations {pose estimation}.

scene reconstruction
Using several scene images, calculate three-dimensional model {scene reconstruction}.

structure from motion
Motions cause disparities and disparity rates, which can reveal structure {structure from motion}. Bundle-adjustment algorithms can find three-dimensional scene structure and camera trajectories. First part {projective reconstruction} can construct projected structure, and second part {Euclidean upgrading} can find actual shape. Affine reconstruction uses Tomasi-Kanade factorization.

MATH>Computer Science>Systems>Evolution Theories

evolution theories
Natural selection can work on differences to select optima {evolution theories}.

adaptive walk
In fitness landscapes, starting from any genotype, change one allele randomly to go to neighboring genotype, and then stay there if it has higher fitness {adaptive walk}|. Adaptive walks go uphill to local peaks.
steps
After uphill steps, number of possibly higher steps becomes half or less and finding a higher step takes twice as long or longer.
If starting at low fitness, large steps work best. If starting at high fitness, small steps work best.
recombination
Recombination allows leaving local maximum to look for global maximum, if landscape is not too random.
few alleles
With two alleles for genes, number of steps to local peak is natural logarithm of gene number. Many local peaks are nearby. Because changes are random and neighboring fitnesses do not correlate, selection makes little improvement.
multiple gene changes
If several alleles change, steps go farther, and fitness-change range is greater.
gene number
If gene number increases, fitness increase per step is less.

coupled fitness landscape

Gene fitness can depend on other-gene fitness in same or other species {coupled fitness landscape} {fitness landscape, coupled}.

If species dependency is low, same-species gene dependency is high, or species number is low, system is stable. If species dependency is high, same-species gene dependency is low, or species number is high, system is unstable.

If species number or species-dependency level can change, species dependency tends to middle, to maximize average fitness and minimize average extinction rate. Middle level is just before point where too many changes happen.

fitness in evolution

Genes have alleles, which contribute to survival and reproduction {fitness, allele}.

fitness landscape

Graphs {fitness landscape, evolution} {genetic graph} show all genes and alleles in a genotype. Genotypes have height. Genotypes differ from neighboring genotypes by one allele.

correlations

Fitness landscapes can have neighbors with similar fitness {correlated fitness}, so allele fitness depends on other-gene fitness. If allele fitness greatly depends on other genes, allele change affects whole system, fitnesses are similar, and landscape has many local peaks. If genes affect all others, fitnesses are equal, as in random case. If allele fitness depends slightly on other genes, allele change affects small region, fitnesses are different, and landscape has few local peaks. Highest peaks have widest bases.

patch

Genetic graphs can divide into local maximum-fitness regions. Patch boundaries are at local fitness minima.

search

In fitness landscapes, search for maximum can stop at local maximum. To escape local maximum or minimum, methods can look at patches or ignore input from neighbors. Instead of moving from fitness-landscape point to point, trajectories can move from patch to patch, seeking highest maximum. If patch size is small, system takes many steps to reach maximum fitness, can move away from true maximum, and can change over time, so it never reaches maximum fitness. If patch size is large, patches can have several maximum-fitness peaks. Moderate patch size balances fitness and time. If fitness landscape is smooth, patches have no affect.

Systems can ignore input from neighbors and use input from farther away. If jumps are small, system takes many steps to reach maximum fitness, can move away from true maximum, and can change over time, so it never reaches maximum fitness. If jumps are large, system can jump several maximum-fitness peaks. Moderate jump size balances fitness and time.

genetic algorithm

Systems {genetic algorithm} can perform similar processes varying by one or more parameters, in response to environment situations that require learning. Some processes are more correct or successful and have better fitness. More-successful processes can continue by selection and/or initiate similar processes by reproduction. Systems thus move toward higher percentage of more-successful processes. Systems have evolution.

genetic circuit system

In Boolean systems {genetic circuit, mathematics} {genetic network model}, repressors can turn off gene expression. Enhancers can turn on gene expression. Genes are on or off. Repressors and enhancers are either present or absent. Boolean canalyzing functions model gene regulation. Without repressor, gene expresses. With derepressor, gene expresses. Different repressor or derepressor amounts can modulate gene expression. Promoter, derepressor, or other gene sites can regulate gene expression.

MATH>Computer Science>Systems>Nanotechnology

nanotechnology

Manipulating molecules and atoms {nanotechnology}|, such as DNA, RNA, or protein strands, can create machines and programs. Cross-links and ratchets can make rotations and slides.

bis-peptide

Amino-acid-like compounds {bis-peptide} {bis-amino acid} have two pairs, each with carboxyl and amino, on opposite sides for hexagon shape, 90-degree sides for square shape, or 60-degree sides for triangle shape, that can link to make peptide-like structures with different shapes.

MATH>Computer Science>Systems>Self-Organization

self-organization
Physical-unit interactions can order systems {self-organization}| {order for free}. Forces and interactions among individuals can result in complexes.
dissipative system
Open non-equilibrium thermodynamic systems input and output matter and energy. Examples are cells and organisms. Dissipative systems have no general laws. If systems have diverse elements that can interact to make new things in closed space, catalytic things can increase original reaction rates {autocatalytic system, self-organization}. System components can interact and organize into patterned subsystems by relaxation methods. Self-organization involves compensations and rearrangements among group parts. Interactions and patterns can cause groups to exhibit behavior not found in parts. Life exists between chaos and order, as at phase-transition boundary, because variation must not be too little or too much.

self-organized criticality
Systems that add matter and energy can be in states {self-organized criticality} in which catastrophe strikes large and small, in power-law relations.

MATH>Computer Science>Internet

Web
Publicly accessible computer networks {Web} {World Wide Web}| can use input/output communication lines.

internetworking
Different local networks can combine into one network {internetworking}.

client computer
Servers {server computer} route messages between clients {client computer}. Clients and servers have physical local network addresses {Media Access Control} (MAC), physical global network addresses {IP address}, network names, relative locations to other local servers or clients, and cryptographic key names.

web page
Web sites have one or more HTML files {web page}|.

robots.txt file
At web-site root level, files {robots.txt file} can instruct robots to disallow directories.

MATH>Computer Science>Internet>Naming

domain name
Web sites have names {domain name}|, such as "earthlink.com" or "menten.info", which people must register with ICAAN. Domains can have subdomains, such as "mail.earthlink.com" or "music.earthlink.com". Domain names are on networks, such as "www", as in "www.earthlink.com".

Uniform Resource Identifier
Web sites have addresses {Uniform Resource Locator} (URL) {Uniform Resource Identifier} (URI), which typically include protocol type, such as "http", network, such as "www", and domain name, such as "earthlink.com", making URI "http://www.earthlink.com". Web sites typically have "index.html" index file, which appear by default, in directories below web site, so URI is "http://www.earthlink.com/" or "http://www.earthlink.com/index.html".

website
Computers can have publicly accessible directory with index file in HTML format {website}|, which other computers can access over network. Web sites can use CGI tools, such as guestbook, hit counter, and message board. Web sites can allow FTP or have FTP tools, such as Fetch. Web sites can allow databases, such as MySQL. Web sites can allow password protection. Web sites can allow SSL encryption. Web sites have allotted disk space, such as 2 GB or 400 MB. Web sites have allowed data transfer amounts each month, such as 20 GB.

MATH>Computer Science>Internet>Programs

agent on Internet
Web sites can have programs {agent, Internet}| {intelligent agent, Internet} that aid selecting among choices, filling forms, or searching.

common gateway interface
On HTTP servers, Perl or C++ programs and scripts can use interfaces {common gateway interface}| with common functions.

search engine
Web-site programs {search engine}| can look in indexed databases of stored web pages or other information to find logical matches to entered words or phrases.

MATH>Computer Science>Internet>Programs>Robots

robot on Internet
Web-site programs {robot, Internet}| {spider, Internet} {crawler, Internet} {web crawler} {web wanderer} {wanderer, Internet} can automatically follow HREF referenced site links on web pages and download found web pages for indexing, validating HTML and/or links, noting what is new, or mirroring sites.

ant on Internet
Robots {ant, Internet}| {web ant} on different web pages and servers can cooperate.

MATH>Computer Science>Internet>Programs>Intrusive

cookie on Internet
Web-page servers can place small files {cookie, Internet}| on client-computer hard disks, to note that client accessed web page. Cookie files contain unique identifiers, so server can track computer activity.

web beacon
Web-page programs {web beacon}| can record number of times that computer accessed page or cookie.

MATH>Computer Science>Internet>Programs>Intrusive>Malware

phishing
Web sites, emails, or text messages can ask for private information {phishing}|, using pretexts.

spyware
Web-site programs {spyware}| can report user information to another web site.

trojan horse on Internet
Web-site programs {trojan horse, Internet}| can seem OK but have malicious code.

worm on Internet
Web-site programs {worm, Internet}| can follow links on web pages and try to replicate on web-page servers.

MATH>Computer Science>Internet>Protocol

protocol on Internet
Accessing web sites uses method {protocol, Internet}|, such as "http" or "ftp".

end-to-end principle
Internet hardware and software {Internet Protocol} should not affect network behavior {end-to-end principle}.

MATH>Computer Science>Internet>Levels

internet-0

All devices can interconnect {internet-0}.

internet-1

Current Internet connects computers {internet-1}. Software generates communication-line signals, encodes and decodes, sends and receives packets, assembles and disassembles packets, and standardizes packet contents.

internet-2

Internet can be high-speed {internet-2}.

bit size on Internet

Speed times sending time can be bigger or smaller than network size {bit size}. If bit size is bigger, there is no overlap, so network does not need special hubs, cables, and transceivers to coordinate different amplitudes, frequencies, and phases. System has no reflections or refractions. At gigabit-per-second speeds, in Internet, bit size is 30 centimeters. At megabit-per-second speeds, in home networks, bit size is 300 meters.