PHILOSOPHY
OF COMPUTER SCIENCE

CD5650

COMPUTABILITY

Gordana Dodig-Crnkovic

Department of Computer Science and Engineering
Mälardalen University, 23 January 2004

CONTENT

PART I

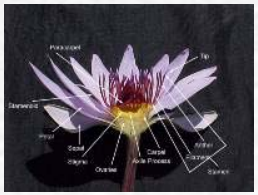
PROLOGUE

A Few Meta-Level Words
or
Lecture Use Instruction

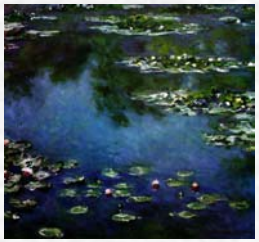
"Real" Waterlilies

"Real" Waterlilies

"Real" Waterlilies



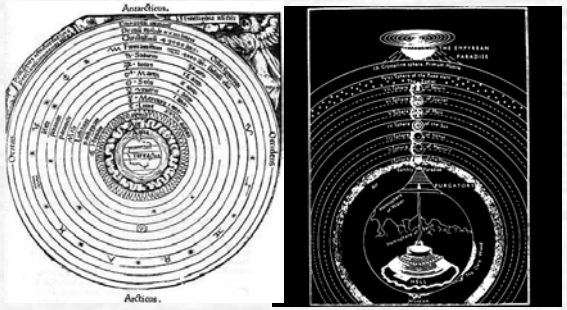
Claude Monet: Blue Waterlilies
Are They Real?



This lecture is more like an impressionist painting giving you
a general ideas with a very few details!

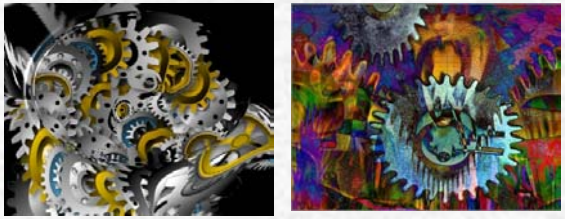
THE UNIVERSE AS A COMPUTER

The Medieval Universe with Earth in the Centre



From Aristotle *Libri de caelo* (1519).

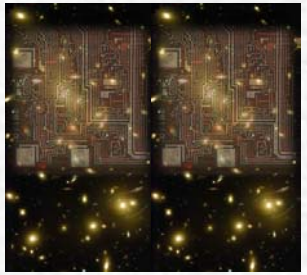
The Clockwork Universe



The mechanistic paradigm which systematically revealed physical structure in analogy with the artificial.

The self-functioning automaton - basis and canon of the form of the Universe.
Newton *Principia*, 1687

THE UNIVERSE AS A COMPUTER



We are all living inside a gigantic computer. No, not The Matrix: the Universe.

Every process, every change that takes place in the Universe, may be considered as a kind of computation.

E Fredkin, S Wolfram

http://www.nature.com/nsu/020527/020527-16.html

THE WILDFIRE SPREAD OF COMPUTATIONAL IDEAS

"Everyone knows that computational and information *technology* has spread like wildfire throughout academic and intellectual life. But the spread of computational *ideas* has been just as impressive.

Biologists not only model life forms on computers; they treat the gene, and even whole organisms, as *information systems*. Philosophy, artificial intelligence, and cognitive science don't just construct computational models of mind; they take cognition *to be computation*, at the deepest levels.

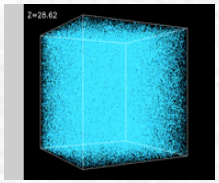
THE WILDFIRE SPREAD OF COMPUTATIONAL IDEAS

Physicists don't just talk about the information carried by a subatomic particle; they propose to unify the foundations of quantum mechanics *with notions of information*.

Similarly for linguists, artists, anthropologists, critics, etc. Throughout the university, people are using computational and information notions -- such as information, digitality, algorithm, formal, symbol, virtual machine, abstraction, implementation, etc. -- as fundamental concepts in terms of which to formulate their theoretical claims."

Brian Cantwell Smith, 2003

THE UNIVERSE AS A COMPUTER



String formation – Andrei Linde

http://physics.stanford.edu/linde

A simulation of large-scale structure formation

http://cfcp.uchicago.edu/lss/sims.html

[**Ontology** What may be known about what may exist.]

PHILOSOPHICAL PROBLEMS OF COMPUTING

WHAT IS COMPUTING? WHAT IS COMPUTER?

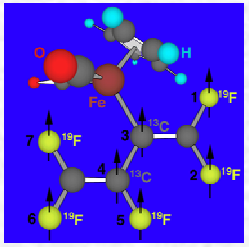
Quantum Computer

IBM's quantum computer uses the interactions of nuclear spins within a specially designed molecule to perform calculations in a manner that is exponentially more powerful than conventional computers.

The spins are programmed by a series of radiofrequency pulses and the answer is read from a nuclear magnetic resonance spectrum.

http://domino.research.ibm.com/comm/bios.nsf/pages/quantum.html

WHAT IS COMPUTING? WHAT IS COMPUTER?

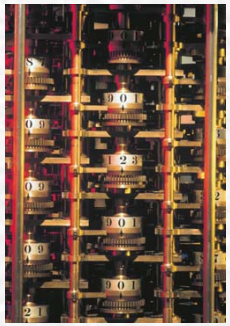


This molecule is currently the world's most advanced quantum computer - a 7-qubit quantum that IBM researchers used to conduct the first demonstration of Shor's quantum factoring algorithm.

Each of the five fluorine and two carbon-13 atoms in this molecule can act as a quantum bit, or qubit, to solve mathematical problems because their spins can interact with each other as well as be individually programmed (by radiofrequency pulses) and detected (by nuclear magnetic resonance).

http://domino.research.ibm.com/comm/bios.nsf/pages/quantum.html

WHAT IS COMPUTING? WHAT IS COMPUTER?



Babbage's Difference Engine No 1, 1832. Front detail.

Science Museum London

WHAT IS COMPUTING? WHAT IS COMPUTER?

Code-breaking personnel at Bletchley Park, 1943.

This shows one of the Hut 3 priority teams at Bletchley Park, in which civilian and service personnel worked together at code- breaking.

WHAT IS COMPUTING? WHAT IS COMPUTER?

The computer presents itself as a culturally defining technology and has become a symbol of the new millennium, playing a cultural role *far more influential than the mills in the Middle Ages, mechanical clocks in the seventeenth century, or the steam engine in the age of the industrial revolution*.

(Bolter 1984)

PHILOSOPHY OF COMPUTING OR PHILOSOPHY OF INFORMATION*?

DICHOTOMY

INFORMATION – COMPUTATION
SUBSTANTIVE - VERB
DATA STRUCTURE – FUNCTION/ALGORITHM
PARTICLE – FORCE (FIELD)

Instructive analogy from physics:
PARTICLES are considered as the primary principle.
FIELDS/INTERACTIONS are defined in terms of particles, particle exchange.

*What 's in a name? That which we call a rose by any other name would smell as sweet.
William Shakespeare (1564–1616), *Romeo and Juliet. Act ii. Sc. 2.* 1.

WHAT IS INFORMATION?

Luciano Floridi

INFORMATION

There is no consensus yet on the definition of semantic information.

The Standard Definition of declarative, objective and semantic Information (SDI):

information = meaningful data

Floridi's main thesis is that meaningful and well-formed data constitute information only if they also qualify as contingently truthful.

THE PHILOSOPHY OF INFORMATION (PI)

A new philosophical discipline, concerned with

a) the critical investigation of the conceptual nature and basic principles of information, including its dynamics (especially computation and flow), utilisation and sciences; and

b) the elaboration and application of information-theoretic and computational methodologies to philosophical problems.

L. Floridi

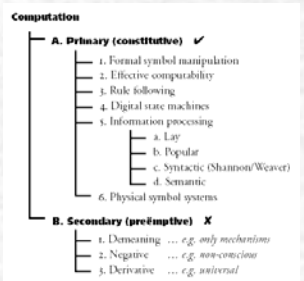
"What is the Philosophy of Information?", Metaphilosophy, 2002
http://www.wolfson.ox.ac.uk/~floridi/papers.htm

WHAT IS COMPUTATION?

Brian Cantwell Smith

CONSTRUALS OF COMPUTATION

Brian Cantwell Smith
The Age of Significance



CONSTRUALS OF COMPUTATION

1. **Formal symbol manipulation**
   the idea, derivative from a century's work in formal logic and meta-mathematics, of a machine manipulating symbolic or meaningful expressions without regard to their interpretation or semantic content;
   Calculation of a function behavior that, when given as input an argument to a (typically mathematical) function, produces as output the value of that function on that argument;

2. **Effective computability**
   what can be done mechanically, as it were, by, an abstract analogue of a "mere machine";

3. **Rule-following or algorithm execution**
   what is involved, and what behavior is thereby produced, in following a set of rules or instructions, such as when cooking dessert;

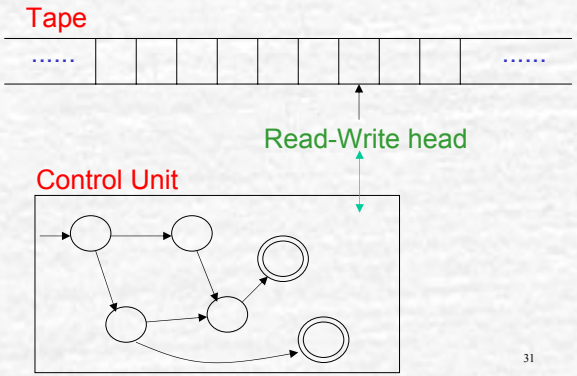
4. **Digital state machines**
   the idea of an automaton with a finite, disjoint set of internally homogeneous states;

5. **Information processing**
   what is involved in storing, manipulating, displaying, and otherwise trafficking in "information," whatever information might be; and

6. **Physical symbol systems**
   the idea, made famous by Newell and Simon, that, somehow or other, computers interact with and perhaps are also made of symbols in a way that depends on their mutual physical embodiment.

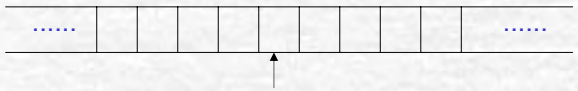
TURING MACHINES

A Turing Machine



The Tape

No boundaries -- infinite length

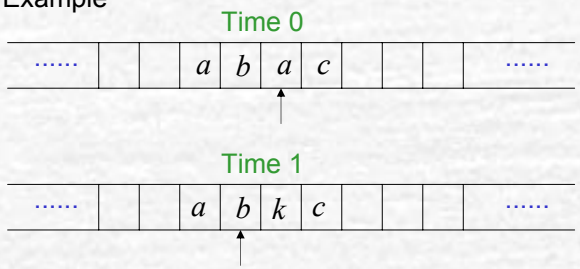


The head moves Left or Right

The head at each time step:

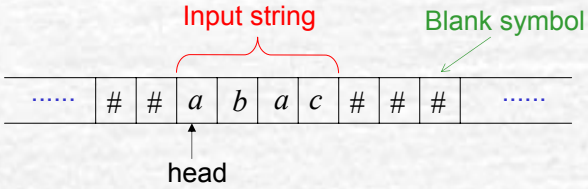
1. Reads a symbol
2. Writes a symbol
3. Moves Left or Right

Example



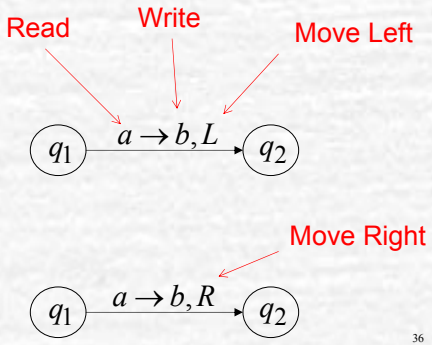
1. Reads a
2. Writes k
3. Moves Left

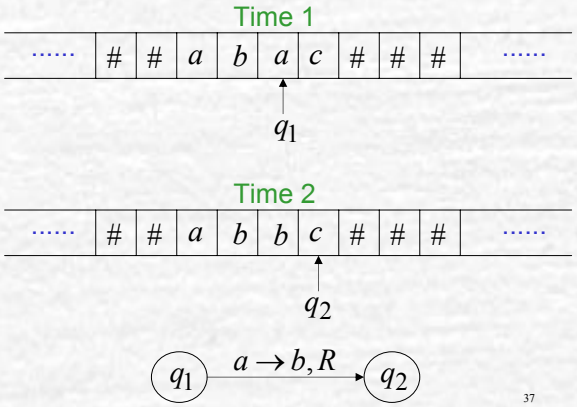
The Input String



Head starts at the leftmost position of the input string

States & Transitions

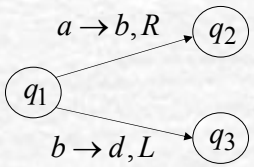




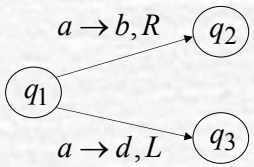
Determinism

Turing Machines are deterministic

Allowed



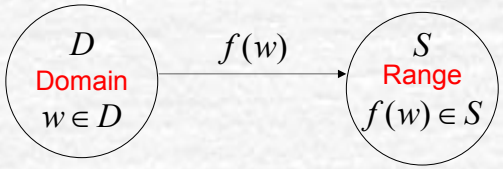
Not Allowed



No lambda transitions allowed in TM!

COMPUTING FUNCTIONS WITH TURING MACHINES

A function



A function may have many parameters:

Example: Addition function

$$f(x, y) = x + y$$

Integer Domain

Decimal: 5

Binary: 101

Unary: 11111

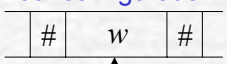
We prefer unary representation:

easier to manipulate

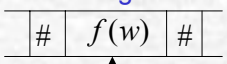
Definition

A function f is computable if there is a Turing Machine M such that:

Initial configuration



Final configuration



For all $w \in D$ Domain

Example (Addition)

The function $f(x, y) = x + y$ is computable

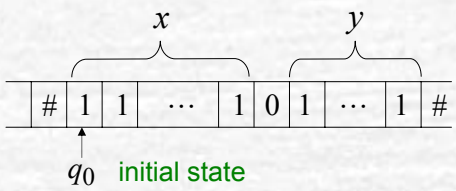
x, y are integers

Turing Machine:

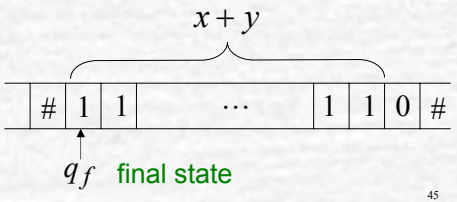
Input string: $x0y$ unary

Output string: $xy0$ unary

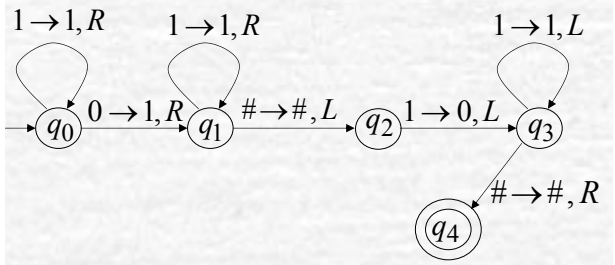
Start



Finish



Turing machine for function $f(x,y)=x+y$

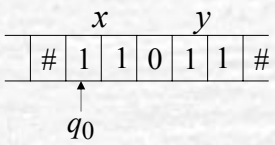


Execution Example:

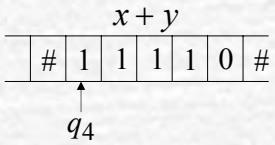
$x=11$ (2)

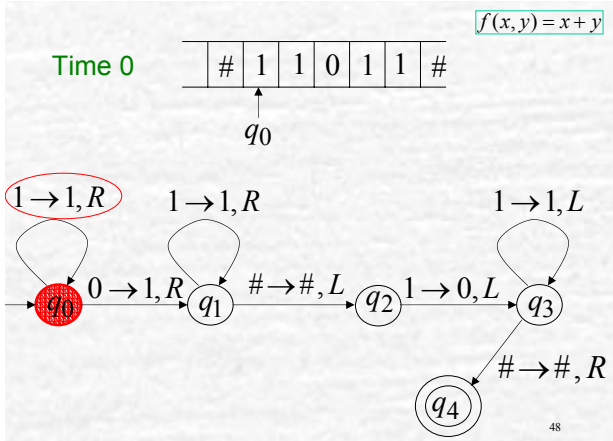
$y=11$ (2)

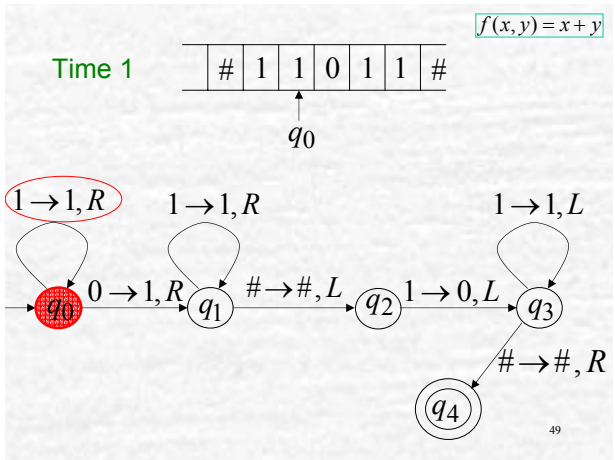
Time 0

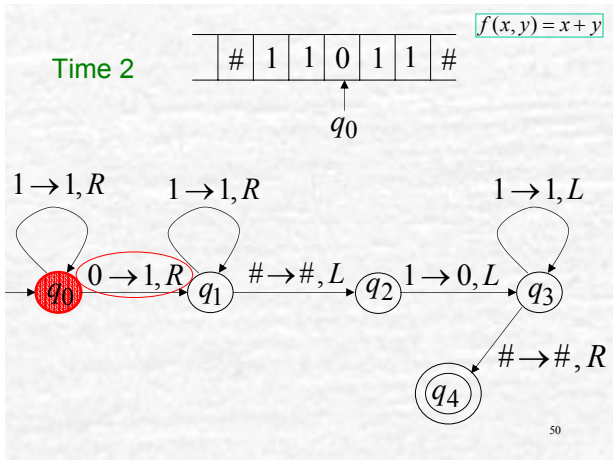


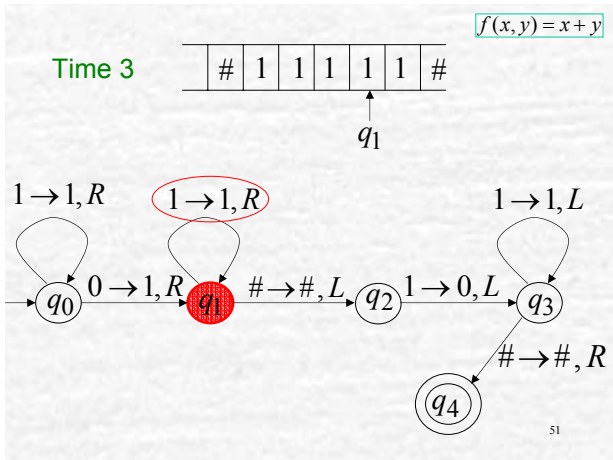
Final Result

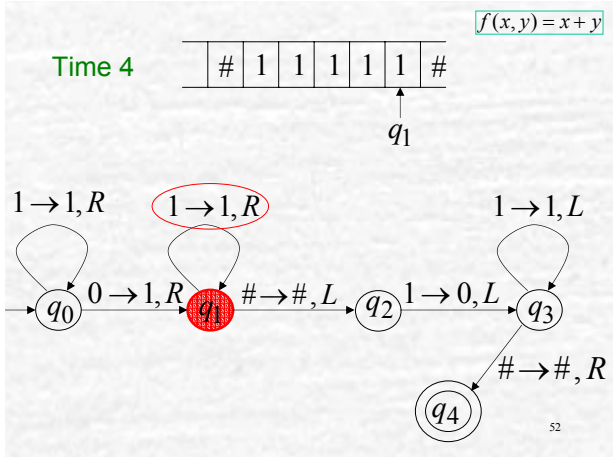


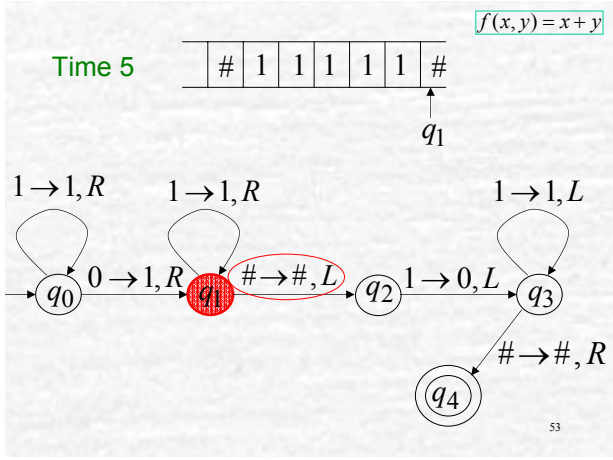


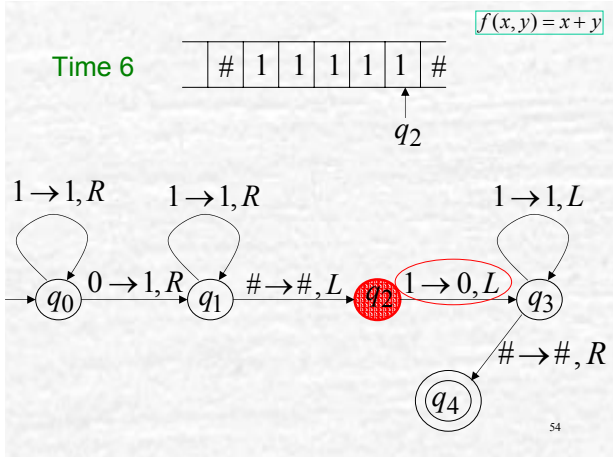


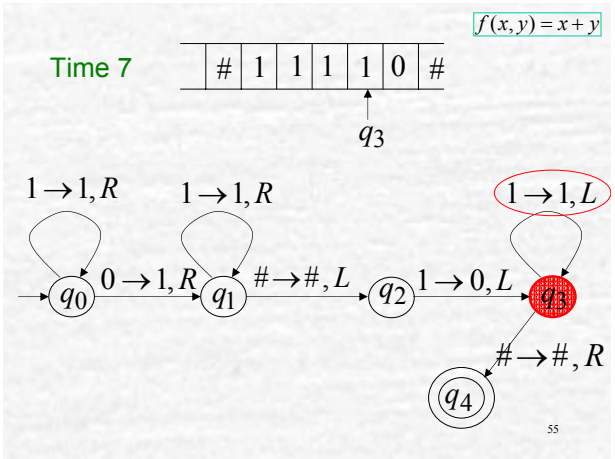


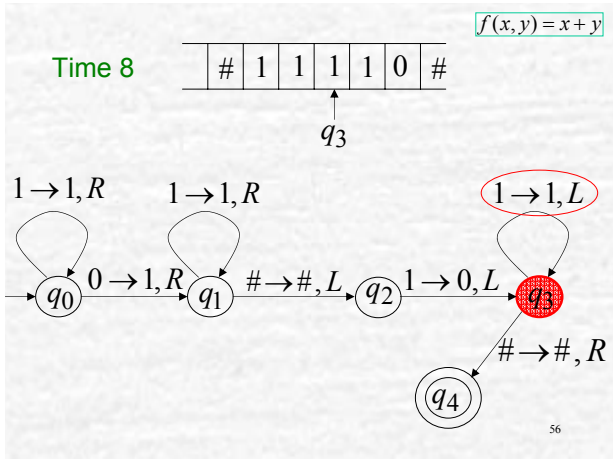


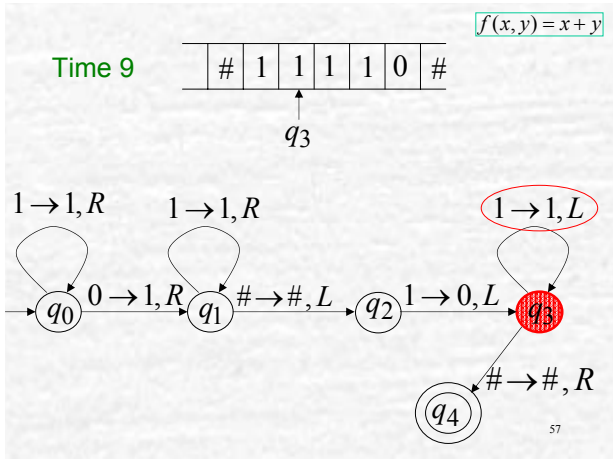


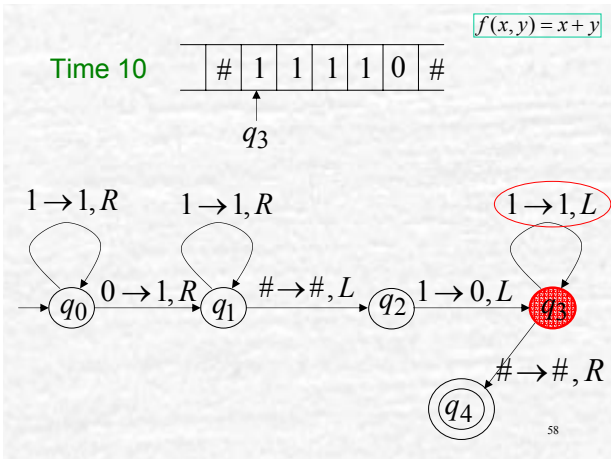


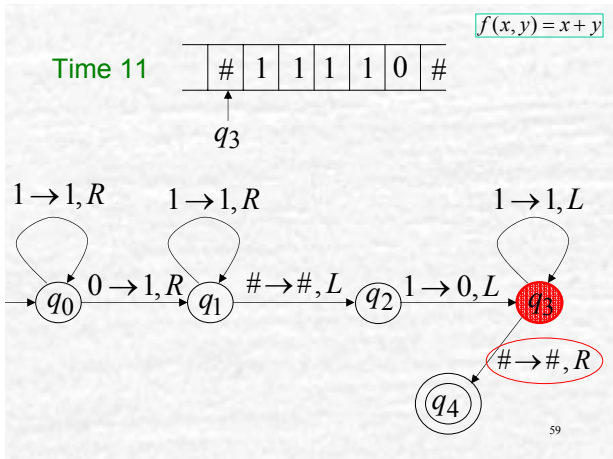


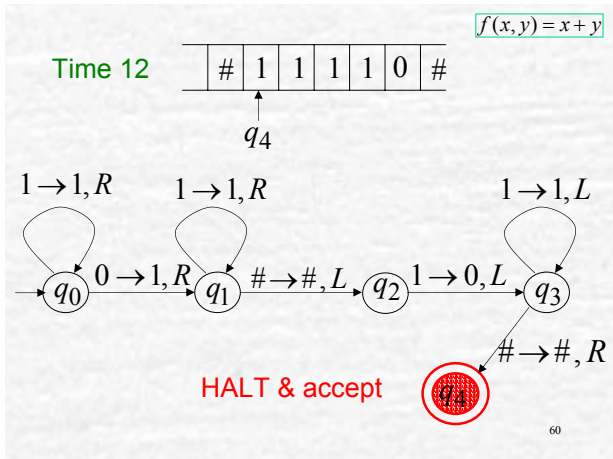








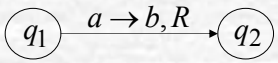




Formal Definitions for Turing Machines

61

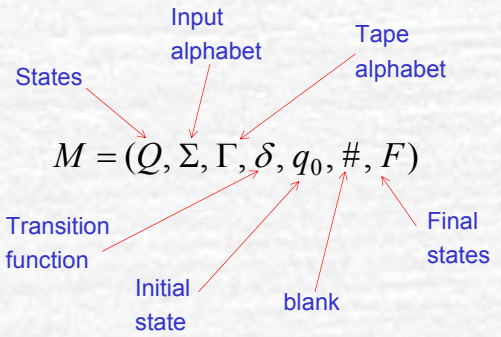
Transition Function



$$\delta(q_1,a)=(q_2,b,R)$$

62

Turing Machine



63

PART II

UNIVERSAL TM'S; DECIDABILITY
UNCOMPUTABLE FUNCTIONS
HILBERT'S PROGRAM [AND GÖDEL THEOREM]
TURING THESIS; CHURCH-TURING THESIS

UNIVERSAL TURING MACHINE

A limitation of Standard Turing Machines:

A Standard Turing Machine is "hardwired"

it executes
only one program

Solution: Universal Turing Machine

Characteristics:

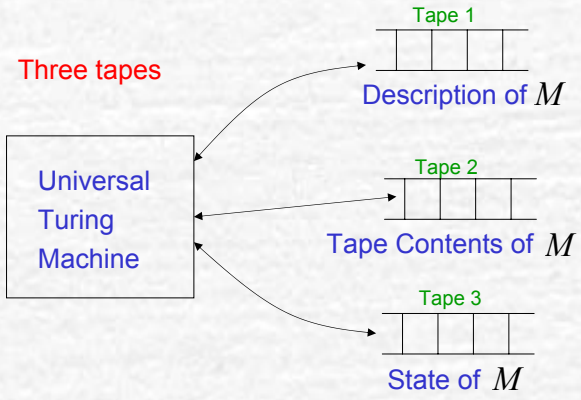
- Reprogrammable machine
- Simulates any other Turing Machine

Universal Turing Machine

Input of Universal Turing Machine

- Description of transitions of M
- Initial tape contents of M

Three tapes



Description of M

We describe Turing machine M
as a string of symbols:

We encode M as a string of symbols

Alphabet Encoding

Symbols:	a	b	c	d	$\cdots$
Encoding:	1	11	111	1111	

State Encoding

States:	q_1	q_2	q_3	q_4	$\cdots$
Encoding:	1	11	111	1111	

Head Move Encoding

Move:	L	R
Encoding:	1	11

Transition Encoding

Transition: $\delta(q_1, a) = (q_2, b, L)$

Encoding: 1 0 1 0 11 0 11 0 1

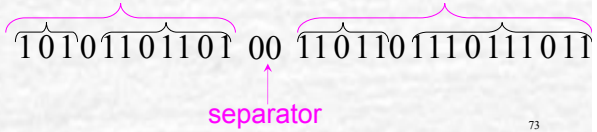
separator

Machine Encoding

Transitions:

$\delta(q_1, a) = (q_2, b, L)$ $\quad$ $\delta(q_2, b) = (q_3, c, R)$

Encoding:



Decidability

A problem is decidable if some Turing machine solves (decides) the problem, i.e. comes up with answer YES or NO.

Decidable problems:

- Does machine M have three states ?
- Is string w a binary number?
- Does DFA* M accept any input?

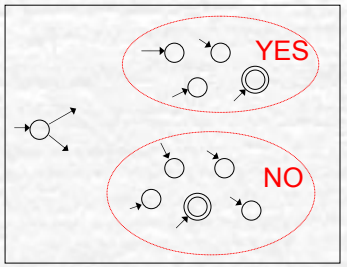
* DFA = Deterministic Finite Automaton

The Turing machine that decides a problem answers YES or NO for each instance.

The machine that decides a problem:

- If the answer is YES then halts in a yes state
- If the answer is NO then halts in a no state

Turing Machine that decides a problem



YES and NO states are halting states

Some problems are undecidable:

There is no Turing Machine that solves all instances of the problem.

A famous undecidable problem:

The halting problem

The Halting Problem

Input: • Turing Machine M
• String w

Question: Does M halt on w?

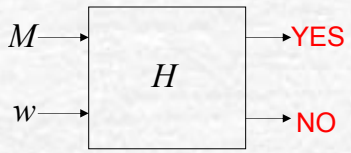
Theorem

The halting problem is undecidable.

Proof (by contradiction)

Assume to the contrary that
the halting problem is decidable.

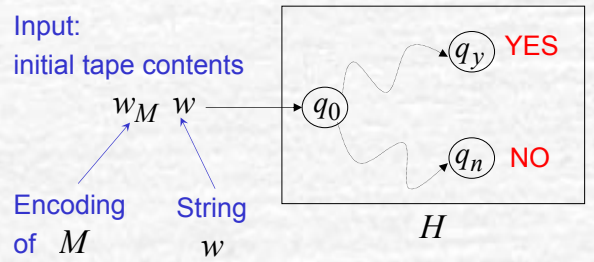
There exists Turing Machine H
that solves the halting problem



M halts on w

Construction of H

Input:
initial tape contents



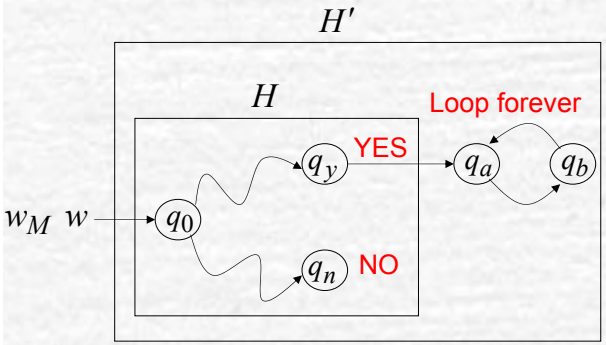
Encoding of M

String w

Construct machine H'

If H returns YES then loop forever.

If H returns NO then halt.



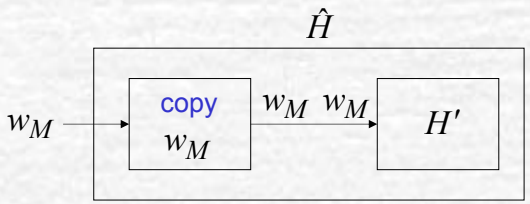
Construct machine $\hat{H}$

Input: w_M (machine M)

If M halts on input w_M

Then loop forever

Else halt



Run machine $\hat{H}$ with input itself

Input: $w_{\hat{H}}$ (machine $\hat{H}$)

If $\hat{H}$ halts on input $w_{\hat{H}}$

Then loop forever

Else halt

$\hat{H}$ on input $w_{\hat{H}}$:

If $\hat{H}$ halts then loops forever.

If $\hat{H}$ doesn't halt then it halts.

CONTRADICTION !

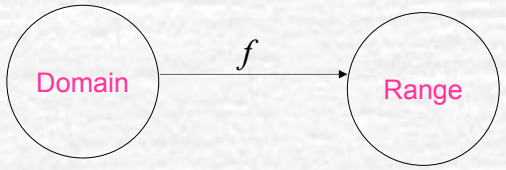
This means that

The halting problem is undecidable.

END OF PROOF

Uncomputable Functions

Uncomputable Functions



A function is uncomputable if it cannot be computed for all of its domain.

Example

An uncomputable function:

$f(n) =$ maximum number of moves until any Turing machine with n states halts when started with the blank tape.

Theorem

Function $f(n)$ is uncomputable.

Proof

Assume to the contrary that $f(n)$ is computable.

If it is so, then the blank-tape halting problem is decidable.

HILBERT'S PROGRAM

HILBERT'S PROGRAM FOR MATHEMATICS

1900 Paris International Congress of Mathematicians (23 mathematical problems for the century to come).

Hilbert's hope was that mathematics would be reducible to finding proofs (manipulating the strings of symbols) from a fixed system of axioms, axioms that everyone could agree were true.

http://aleph0.clarku.edu/~djoyce/hilbert/problems.html
http://mathworld.wolfram.com/HilbertsProblems.html

HILBERT'S PROGRAM FOR MATHEMATICS

Can all of mathematics be made *algorithmic,* or will there always be new problems that outstrip any given algorithm, and so require creative acts of mind to solve?

GÖDEL: TRUTH AND PROVABILITY

Kurt Gödel actually proved two related fundamental theorems. They have revolutionized mathematics, showing that mathematical truth is more than logic and computation.

Gödel has been called the most important logician since Aristotle. His two theorems changed logic and mathematics as well as the way we look at *truth* and *proof.*

GÖDEL: TRUTH AND PROVABILITY

Gödels first theorem proved that any formal system strong enough to support number theory has at least one undecidable statement. Even if we know that the statement is true, the system cannot prove it. This means the system is incomplete. For this reason, Gödel's first proof is called "*the incompleteness theorem*".

GÖDEL: TRUTH AND PROVABILITY

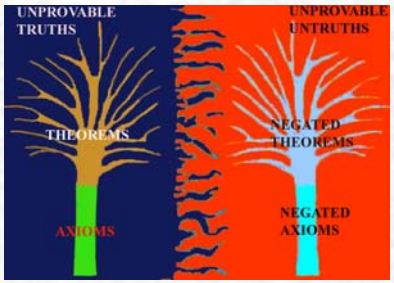
Gödel's second theorem is closely related to the first. It says that no one can prove, from inside any complex formal system, that it is self-consistent.

GÖDEL: TRUTH AND PROVABILITY

"Gödel showed that provability is a weaker notion than truth, no matter what axiomatic system is involved."

In other words, *we simply cannot prove some things* in mathematics (from a given set of premises) *which we nonetheless can know are true*. (D. Hofstadter)

TRUTH VS. PROVABILITY ACCORDING TO GÖDEL



After: Gödel, Escher, Bach - an Eternal Golden Braid by Douglas Hofstadter.

TRUTH VS. PROVABILITY ACCORDING TO GÖDEL

A Post Script

Gödel theorem is built upon Aristotelian logic.

So it is true within the paradigm of Aristotelian logic.

CHURCH-TURING THESIS*

*Source: Stanford Encyclopaedia of Philosophy (B. Jack Copeland)

A Turing machine is an abstract representation of a computing device.

It is more like a computer program (software) than a computer (hardware).

LCMs [Logical Computing Machines: Turing's expression for Turing machines] were first proposed by Alan Turing, in an attempt to give a mathematically precise definition of "algorithm" or "mechanical procedure".

A method, M, is called 'effective' or 'mechanical' just in case:

1. M is set out in terms of a finite number of exact instructions (each instruction being expressed by means of a finite number of symbols);
2. M will, if carried out without error, always produce the desired result in a finite number of steps;
3. M can (in practice or in principle) be carried out by a human being unaided by any machinery except for paper and pencil;
4. M demands no insight or ingenuity on the part of the human being carrying it out.

Turing's thesis: LCMs [TMs] can do anything that could be described as "rule of thumb" or "purely mechanical". (Turing 1948)

He adds: This is sufficiently well established that it is now agreed amongst logicians that "calculable by means of an LCM" is the correct accurate rendering of such phrases.

Computer Science Law

A computation is mechanical/effective
if and only if
it can be performed by a Turing Machine.

Definition of Algorithm

An algorithm for function $f(w)$
is a Turing Machine which computes $f(w)$

Algorithms are Turing Machines!

When we say

There exists an algorithm

It means

There exists a Turing Machine.

Turing introduced his thesis in the course of arguing that the Entscheidungsproblem, or decision problem, for the predicate calculus - posed by Hilbert (1928) - is unsolvable.

Church's account of the Entscheidungsproblem

By the Entscheidungsproblem of a system of symbolic logic is here understood the problem to find an effective method by which, given any expression Q in the notation of the system, it can be determined whether or not Q is provable in the system.

The truth table test is such a method for the propositional calculus.

Turing showed that, given his thesis, there can be no such method for the predicate calculus.

Predicate calculus formulas are of a type:

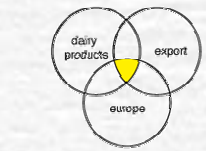
$$\forall x\,(Mx \supset \exists y\,(Ty \,\&\, Dxy))$$

The truth table: AND Operator (&)
(propositional calculus)

dairy products AND export
AND europe

All terms are present

A	B	A&B
0	0	0
0	1	0
1	0	0
1	1	1



AND Gate

Church's thesis: A function of positive integers is effectively calculable only if it is recursive.

PART III

OTHER MODELS OF COMPUTATION
NATURAL COMPUTATION
[BIOLOGICAL COMPUTING, QUANTUM COMPUTING]

CONCLUSIONS

TURING EQUIVALENT (EFFECTIVE) MODELS OF COMPUTATION

- Turing Machines
- Recursive Functions
- Post Systems
- Rewriting Systems
- …

Turing's Thesis

A computation is mechanical if and only if it can be performed by a Turing Machine.

Church's Thesis (extended)

All models of effective computation are equivalent.

Theorem:

A language is *recursively enumerable* if and only if
- a *Turing Machine* / Post system generates it.

REWRITING SYSTEMS

They convert one string to another

- Matrix Grammars
- Markov Algorithms
- Lindenmayer-Systems (L-Systems)

LINDENMAYER-SYSTEMS

They are parallel rewriting systems

Example: $a \rightarrow aa$

Derivation: $a \Rightarrow aa \Rightarrow aaaa \Rightarrow aaaaaaaa$

$$L = \{a^{2^n} : n \geq 0\}$$

Lindenmayer-Systems are not general as recursively enumerable languages

Extended Lindenmayer-Systems: $(x, a, y) \rightarrow u$

context

Theorem:

A language is *recursively enumerable* if and only if an *Extended Lindenmayer-System* generates it.

L-System Example: Fibonacci numbers

Consider the following simple grammar:

variables : A B
constants : none

start: A
rules: A → B
B → AB

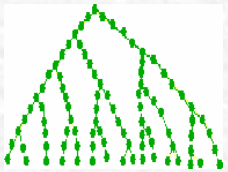
This L-system produces the following sequence of strings ...

Stage 0 : A
Stage 1 : B
Stage 2 : AB
Stage 3 : BAB
Stage 4 : ABBAB
Stage 5 : BABABBAB
Stage 6 : ABBABBABABBAB
Stage 7 : BABABBABABBABBABABBAB

If we count the length of each string, we obtain the Fibonacci sequence of numbers:

1 1 2 3 5 8 13 21 34 ....

Example - Algal growth



The figure shows the pattern of cell lineages found in the alga *Chaetomorpha linum*.

To describe this pattern, we must let the symbols denote cells in different states, rather than different structures.

This growth process can be generated from an axiom A and growth rules

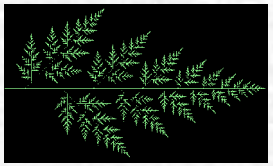
$A \rightarrow DB$
$B \rightarrow C$
$C \rightarrow D$
$D \rightarrow E$
$E \rightarrow A$

Here is the pattern generated by this model.
It matches the arrangement of cells in the original alga.

```
Stage  0 :              A
Stage  1 :           D     B
Stage  2 :           E     C
Stage  3 :           A     D
Stage  4 :        D     B     E
Stage  5 :        E     C     A
Stage  6 :        A     D     D     B
Stage  7 :     D     B     E     E     C
Stage  8 :     E     C     A     A     D
Stage  9 :     A     D  D  B  D  B     E
Stage 10 :  D  B  E  E  C  E  C     A
Stage 11 :  E  C  A  A  D  A  D  D  B
```


EXAMPLE - A COMPOUND LEAF (OR BRANCH)

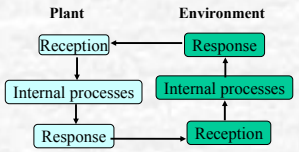
```
Leaf1 {        ; Name of the l-system, "{" indicates start
               ; Compound leaf with alternating branches,
   angle 8       ; Set angle increment to (360/8)=45 degrees
   axiom x       ; Starting character string
   a=n           ; Change every "a" into an "n"
   n=o           ; Likewise change "n" to "o" etc ...
   o=p
   p=x
   b=e
   e=h
   h=j
   j=y
   x=F[+A(4)]Fy   ; Change every "x" into  "F[+A(4)]Fy"
   y=F[-B(4)]Fx   ; Change every "y" into  "F[-B(4)]Fx"
   F=@1.18F@i1.18
   }           ; final } indicates end
```

http://www.xs4all.nl/~cvdmark/tutor.html
(Cool site with animated L-systems)

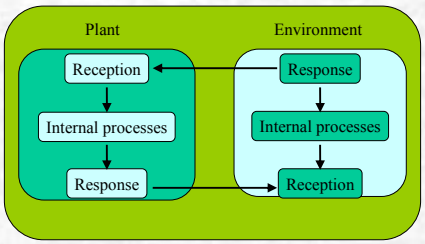
Here is a series of forms created by slowly changing the angle parameter. **lsys00.ls**

Check the rest of the Gallery of L-systems:
http://home.wanadoo.nl/laurens.lapre/

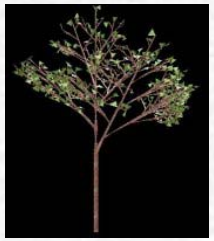


A model of a horse chestnut tree inspired by the work of Chiba and Takenaka.

Here branches compete for light from the sky hemisphere. Clusters of leaves cast shadows on branches further down. An apex in shade does not produce new branches. An existing branch whose leaves do not receive enough light dies and is shed from the tree. In such a manner, the competition for light controls the density of branches in the tree crowns.



Apropos adaptive reactive systems:
"What's the color of a chameleon put onto a mirror?" -Stewart Brand
(Must be possible to verify experimentally, isn't it?)



NATURAL COMPUTATION

COMPUTATION IN PHYSICAL AND BIOLOGICAL SYSTEMS

Computation and information processing may be studied in physical and biological systems that are different from the operations performed by electronic computers.

COMPUTATION IN PHYSICAL AND BIOLOGICAL SYSTEMS

The goal is both of building better electronic computers, by importing strategies used in other devices, and of furthering our understanding of natural processes, by using information-processing principles to explain their behavior.

COMPUTATION IN PHYSICAL AND BIOLOGICAL SYSTEMS

Principles of computation in biological and physical systems have a different character from that of present-day electronic computers.

For example, biological systems are massively parallel and distributed, they use disposable components, they are robust to perturbations in their environment (as discussed earlier), they learn innovative solutions in response to problems, and their global structure and behavior is not directly predictable by simple inspection.

COMPUTATION IN PHYSICAL AND BIOLOGICAL SYSTEMS

Other kinds of physical systems share many of these properties, depending on what level we choose to model them (e.g., quantum, molecular, chemical, or ecosystem).

HOW DOES NATURE COMPUTE?

Relevant questions:

- *How is information processing embedded in dynamical behavior?*
- *How can we detect and then quantify structure in natural processes?*

In pursuing answers to this sort of question we've come to the conclusion that the diverse model classes found in computation theory are key tools in being explicit about how natural information processing mechanisms can be represented and analyzed.

http://www.santafe.edu/sfi/research/focus/compPhysics/ (The Santa Fe Institute)

HOW DOES NATURE COMPUTE?

"However, we also have come to the conclusion that contemporary notions of ``computation'' and of ``useful'' information processing --- colored as they are by the recent history of digital computer technology --- must be extended in order to be useful within empirical science.

Why?

Because the processes studied by natural scientists involve systems that are continuous, stochastic, spatially extended, or some combination of these and other characteristics that fall strictly outside the purview of discrete computation theory. "

http://www.santafe.edu/sfi/research/focus/compPhysics/

BIOLOGICAL COMPUTING

DNA BASED COMPUTING

Despite their respective complexities, biological and mathematical operations have some similarities:

The very complex structure of a living being is the result of **applying simple operations** to initial information encoded in a DNA sequence (genes).

All complex math problems can be **reduced to simple operations** like addition and subtraction.

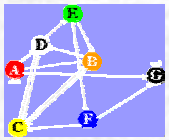
For the same reasons that DNA was presumably selected for living organisms as a genetic material, its stability and predictability in reactions.

DNA strings can also be used to encode information for mathematical systems.

THE HAMILTONIAN PATH PROBLEM

(a "key into lock" problem)

The objective is to find a path from start to end going through all the points only once.



This problem is difficult for conventional (serial logic) computers because they must try each path one at a time. It is like having a whole bunch of keys and trying to see which fits a lock.

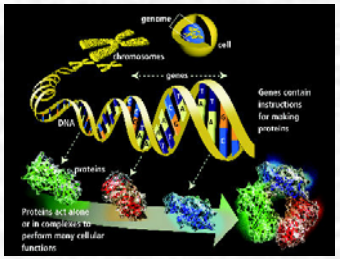
DNA based computers can try all the keys at the same time (massively parallel) and thus are very good at key-to-lock problems, but much slower at simple mathematical problems like multiplication.

The Hamiltonian Path problem was chosen because **every key-to-lock problem can be solved as a Hamiltonian Path problem.**

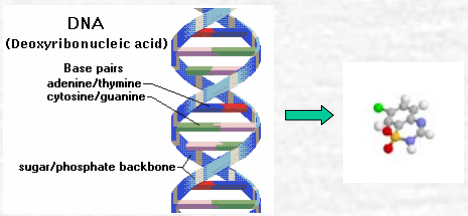
SOLVING THE HAMILTONIAN PATH PROBLEM

1. Generate random paths through the graph.
2. Keep only those paths that begin with the start city (A) and conclude with the end city (G).
3. Because the graph has 7 cities, keep only those paths with 7 cities.
4. Keep only those paths that enter all cities at least once.
5. Any remaining paths are solutions.

DNA



DNA – BASE MOLECULE



SOLVING THE HAMILTONIAN PATH PROBLEM

The key to solving the problem was using DNA to perform the five steps in the above algorithm.

These interconnecting blocks can be used to model DNA:

SOLVING THE HAMILTONIAN PATH PROBLEM

DNA tends to form long double helices:

The two helices are joined by "bases", represented here by coloured blocks. Each base binds only one other specific base. In our example, we will say that each coloured block will only bind with the same colour. For example, if we only had red blocks, they would form a long chain like this:

Any other colour will not bind with red:

PROGRAMMING WITH DNA

Step 1: Create a unique DNA sequence for each city A through G. For each path, for example, from A to B, create a linking piece of DNA that matches the last half of A and first half of B.

PROGRAMMING WITH DNA

Step 2: Because it is difficult to "remove" DNA from the solution, the target DNA, the DNA which started at A and ended at G was copied over and over again until the test tube contained a lot of it relative to the other random sequences.

PROGRAMMING WITH DNA

Step 3: Going by weight, the DNA sequences which were 7 "cities" long were separated from the rest.

PROGRAMMING WITH DNA

Step 4: To ensure that the remaining sequences went through each of the cities, "sticky" pieces of DNA attached to magnets were used to separate the DNA.

PROGRAMMING WITH DNA

Step 5: All that was left was to sequence the DNA, revealing the path from A to B to C to D to E to F to G.

ADVANTAGES

The above procedure took approximately one week to perform. Although this particular problem could be solved on a piece of paper in under an hour, when the number of cities is increased to 70, the problem becomes too complex for even a supercomputer.

While a DNA computer takes much longer than a normal computer to perform each individual calculation, it performs an enormous number of operations at a time (massively parallel).

DNA computers also require less energy and space than normal computers. 1000 litres of water could contain DNA with more memory than all the computers ever made, and a pound of DNA would have more computing power than all the computers ever made.

THE FUTURE

DNA computing is about ten years old and for this reason, it is too early for either great optimism or great pessimism.

Early computers such as ENIAC filled entire rooms, and had to be programmed by punch cards. Since that time, computers have become much smaller and easier to use.

Just as DNA cloning and sequencing were once manual tasks, DNA computers will also become automated.

In addition to the direct benefits of using DNA computers for performing complex computations, some of the operations of DNA computers are used in molecular and biochemical research.

http://www.cis.udel.edu/~dna3/DNA/dnacomp.html ; http://dna2z.com/dnacpu/dna.html;
http://www.liacs.nl/home/pier/webPagesDNA ;
http://www.corninfo.chem.wisc.edu/writings/DNAcomputing.html ;
http://www.comp.leeds.ac.uk/seth/ar35 /

QUANTUM COMPUTING

Today: fraction of micron (10^{-6} m) wide logic gates and wires on the surface of silicon chips.

Soon they will yield even smaller parts and inevitably reach a point where logic gates are so small that they are made out of only a handful of atoms.

1 nm = 10^{-9} m

On the sub-atomic scale matter obeys the rules of quantum mechanics, which are quite different from the classical rules that determine the properties of conventional logic gates.

So if computers are to become smaller in the future, new, quantum technology must replace or supplement what we have now.

WHAT IS QUANTUM MECHANICS?

The deepest theory of physics; the framework within which all other current theories, except the general theory of relativity, are formulated. Some of its features are:

Quantisation (which means that observable quantities do not vary continuously but come in discrete chunks or 'quanta').

Interference (which means that the outcome of a quantum process in general depends on all the possible histories of that process).

This is the feature that makes quantum computers qualitatively more powerful than classical ones.

Entanglement (Two spatially separated and non-interacting quantum systems that have interacted in the past may still have some *locally inaccessible information* in common – information which cannot be accessed in any experiment performed on either of them alone.)

This is the one that makes quantum cryptography possible.

The discovery that quantum physics allows fundamentally new modes of information processing has required the existing theories of computation, information and cryptography to be superseded by their quantum generalisations.

The advantage of quantum computers arises from the way they encode a bit, the fundamental unit of information.

The state of a bit in a classical digital computer is specified by one number, 0 or 1.

An n-bit binary word in a typical computer is accordingly described by a string of *n* zeros and ones.

A quantum bit, called a qubit, might be represented by an atom in one of two different states, which can also be denoted as 0 or 1.

Two qubits, like two classical bits, can attain four different well-defined states (0 and 0, 0 and 1, 1 and 0, or 1 and 1).

But unlike classical bits, qubits can exist simultaneously as 0 and 1, with the probability for each state given by a numerical coefficient.

Describing a two-qubit quantum computer thus requires four coefficients. In general, *n* qubits demand 2^n numbers, which rapidly becomes a sizable set for larger values of n.

For example, if *n* equals 50, about 10^{15} numbers are required to describe all the probabilities for all the possible states of the quantum machine--a number that exceeds the capacity of the largest conventional computer.

A quantum computer promises to be immensely powerful because it can be in multiple states at once (superposition) -- and because it can act on all its possible states simultaneously.

Thus, a quantum computer could naturally perform myriad operations in parallel, using only a single processing unit.

The most famous example of the extra power of a quantum computer is Peter Shor's algorithm for factoring large numbers.

Factoring is an important problem in cryptography; for instance, the security of RSA public key cryptography depends on factoring being a hard problem.

Despite much research, no efficient classical factoring algorithm is known.

However if we keep on putting quantum gates together into circuits we will quickly run into some serious practical problems.

The more interacting qubits are involved the harder it tends to be to engineer the interaction that would display the quantum interference.

Apart from the technical difficulties of working at single-atom and single-photon scales, one of the most important problems is that of preventing the surrounding environment from being affected by the interactions that generate quantum superpositions.

The more components the more likely it is that quantum computation will spread outside the computational unit and will irreversibly dissipate useful information to the environment.

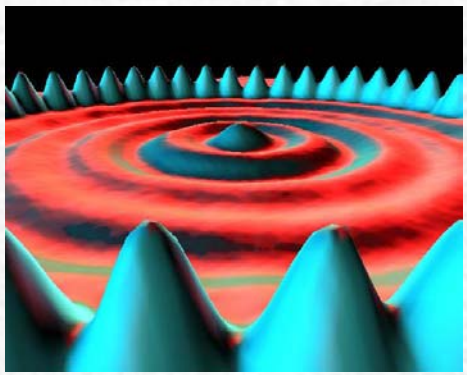
This process is called decoherence. Thus the race is to engineer sub-microscopic systems in which qubits interact only with themselves but not with the environment.

But, the problem is not entirely new!

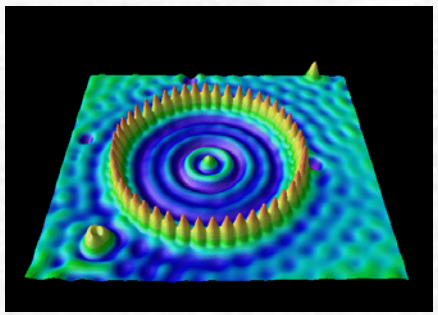
Remember STM?

(Scanning Tuneling Microscopy)

STM was a Nobel Prize winning invention by Binning and Rohrer at IBM Zurich Laboratory in the early 1980s



- **Title** : Quantum Corral
- **Media** : Iron on Copper (111)



The standing-wave patterns in the local density of states of the Cu(111) surface. These spatial oscillations are quantum-mechanical interference patterns caused by scattering of the two-dimensional electron gas off the Fe adatoms and point defects.

WHAT WILL QUANTUM COMPUTERS BE GOOD AT?

The most important applications currently known:

- Cryptography: perfectly secure communication.
- Searching, especially algorithmic searching (Grover's algorithm).
- Factorising large numbers very rapidly (Shor's algorithm).
- Simulating quantum-mechanical systems efficiently

FUNDAMENTAL LIMITS OF COMPUTATION

MISUNDERSTANDINGS OF THE CHURCH-TURING THESIS*

*Based on: The Blackwell Guide to Philosophy of Computing and Information, Chapter 1: 1. Computation: B. Jack Copeland)

MISUNDERSTANDINGS OF THE TURING THESIS

Turing did not show that his machines can solve any problem that can be solved "by instructions, explicitly stated rules, or procedures" and nor did he prove that a universal Turing machine "can compute any function that any computer, with any architecture, can compute".

Turing proved that his universal machine can compute any function that any *Turing machine* can compute; and he put forward, and advanced philosophical arguments in support of, the thesis here called Turing's thesis.

A thesis concerning the extent of effective methods - procedures that a *human being unaided by machinery* is capable of carrying out - has no implication concerning the extent of the procedures that machines are capable of carrying out, even machines acting in accordance with 'explicitly stated rules'.

Among a machine's repertoire of atomic operations there may be those that no human being unaided by machinery can perform.

Turing's "Machines". These machines are *humans* who calculate. (Wittgenstein)

A man provided with paper, pencil, and rubber, and subject to strict discipline, is in effect a universal machine. (Turing)

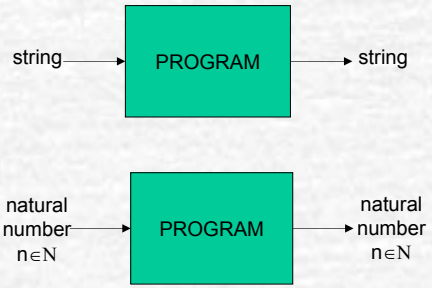
The Entscheidungsproblem is the problem of finding a *humanly executable* procedure of a certain sort, and Turing's aim was precisely to show that there is no such procedure in the case of predicate logic.

CONCLUSIONS

SYMBOLS, STRINGS, PROGRAMS

PROGRAM = STRING (syntactic way)

PROGRAM = FUNCTION $N \to N$ (semantic way)



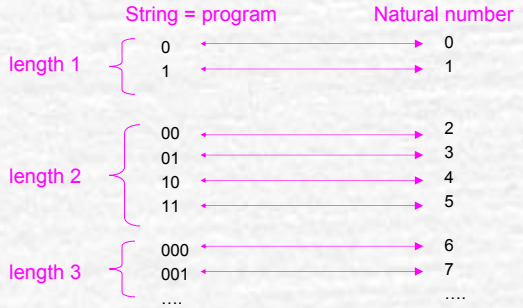
Theorem

The set of all finite strings is countable.

Proof Any finite string can be encoded with a binary string of 0's and 1's

Find an enumeration procedure for the set of finite strings

Produce strings in Proper Order



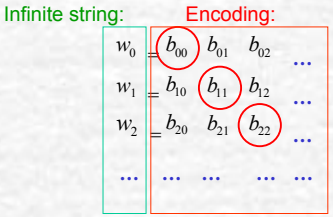
Theorem

The set of all infinite strings is uncountable.

Proof (by contradiction)

We assume we have an enumeration procedure for the set of infinite strings.

CANTOR'S DIAGONAL ARGUMENT



CANTOR'S DIAGONAL ARGUMENT

We can construct a new string w that is missing in our enumeration!

Conclusion

The set of all infinite strings is uncountable!

An infinite string can be seen as FUNCTION $N \to N$ (n:th output is n:th bit in the string)

Conclusion

There are some integer functions that that cannot be described by finite strings (programs/algorithms).

Finite strings (algorithms): countable

Languages (power set of strings): uncountable

There are infinitely many more languages than finite strings.

Conclusion

There are some languages that cannot be described by finite strings (algorithms).

DIFFERENT INFINITIES

- Cardinality of the simplest, ”smallest” infinity (that of a set of natural numbers, e.g.) is $\aleph_0$.
- Cardinality of the set of real numbers, points on a line/plane/body is $\aleph_1$.

REPRESENTATIONAL POWER

Mapping

continuous variable $\aleph_1 \rightarrow$ continuous variable $\aleph_1$

is equivalent to a machine with an infinite symbol set and infinite rule table (which exceeds TM capabilities).

BEYOND THE TURING LIMIT

HYPERCOMPUTATION

HYPERCOMPUTATION

Is computation without an algorithm possible?

The classical concept of an algorithm is a specification of a process that is to take when the algorithm is unrolled into time. [...] One might compare this to the theory of evolution based on natural selection: this is a process-level theory, for which the existence of some *a priori* algorithm is problematic.”

Michael Manthey, Aalborg University in Denmark

HYPERCOMPUTATION

When we observe natural phenomena and we ascribe them computational significance, it is not *the algorithm* we are observing but *the process*, *the computation*.

Hypercomputation means computation without a program.

Some objects might be performing hypercomputation around us: we observe... but we can not describe step-by-step [algorithmically] their computational process.

NEURAL NETWORKS AND ANALOG COMPUTATION - BEYOND THE TURING LIMIT - HAVA SIEGELMANN

Siegelmann-Sontag thesis of

'hypercomputation by analog systems' analogously to the

Church-Turing thesis of

'computation by digital systems'

http://www.cs.umass.edu/~hava/advertisement.html
Neural Networks and Analog Computation: Beyond the Turing Limit

THESIS OF TIME BOUNDED ANALOG COMPUTATION

Any ”reasonable analog computer” will have no more power (up to polynomial speedup) than ARNN (Analog Recurrent Neural Network).

(Siegelmann - Sontag thesis)

NEURAL NETWORKS AND ANALOG COMPUTATION - BEYOND THE TURING LIMIT - HAVA SIEGELMANN

All sets over finite alphabets can be represented as reals that encode the families of Boolean circuits that recognize them. Under efficient time computation, these networks compute not only all efficient computations by Turing machines but also some non-recursive functions such as the halting problem of Turing machines.

Note that while the networks can answer questions regarding Turing machines computation, they still can not answer questions regarding their own halting and computation.

THEME OF THE SECOND AGE - COMPUTING TRANSCENDS COMPUTERS

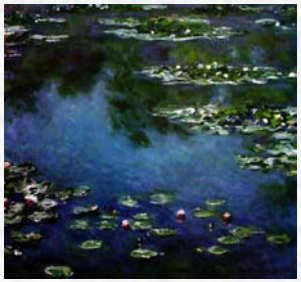
”Everything is up for grabs. Everything will change. There is a magnificent sweep of intellectual landscape right in front of us. ”

David Gelernter, The Second Coming — A Manifesto

http://www.edge.org/3rd_culture/gelernter/gelernter_p1.html

EPILOGUE

After all, this lecture might not be so close to the Blue Waterlilies of Claude Monet (1840-1926) ….



. . . but instead more of a Landscape with Distant River and Bay of another impressionist painter John M William Turner (1775-1851)!