Natural Computing

Dr Giuditta Franco

Department of Computer Science, University of Verona, Italy
giuditta.franco@univr.it

For the logistics of the course, please see the syllabus

Natural Computing

Natural (complex) systems work as (biological) *information elaboration systems*, by sophisticated mechanisms of goal-oriented control, coordination, organization.

Computational analysis (mat modeling) of *life* is important as observing birds for designing flying objects.

"Informatics studies information and computation in natural and artificial systems" (School of Informatics, Univ. of Edinburgh)

Computational processes *observed in* and *inspired by* nature.
Ex. self-assembly, AIS, DNA computing

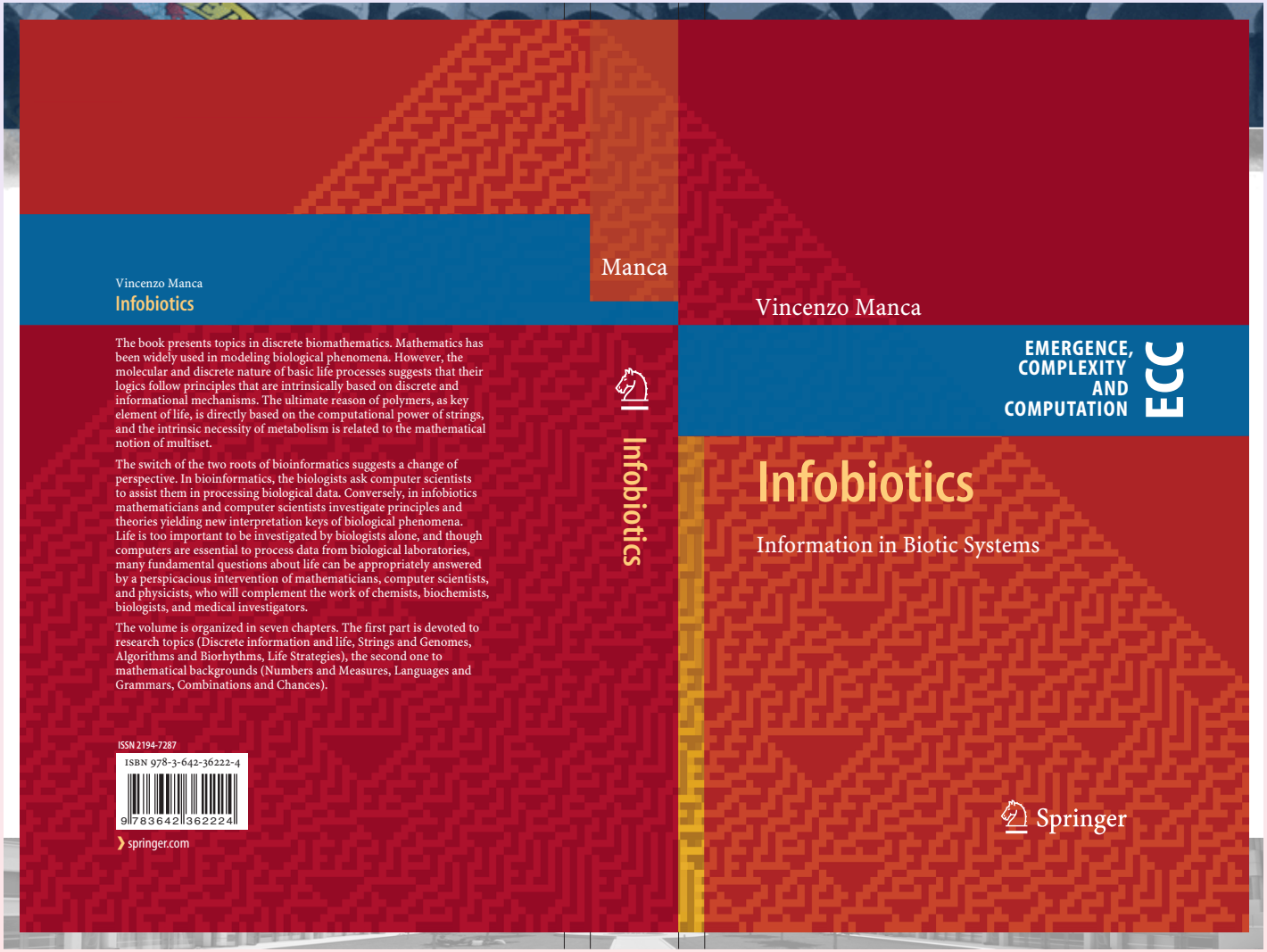
Ex. Internet of things, logics, programming, artificial automata *are not* natural computing.

Bioinformatics versus Infobiotics

Applying statistics, performing tools, high technology, large databases, to analyze bio-logical/medical data, solve problems, formalize/frame natural phenomena. Ex. search algorithms to recover genomic/proteomic data, to process them, catalogue and make them publically accessible, to infer models to simulate their dynamics.

Identifying informational mechanisms underlying living systems, how they assembled and work, how biological information may be defined, processed, decoded. New theories and tools to discover general law underlying biology (mental experiments: Darwin theory, Schrodinger - DNA structure, wave equation, cat experiment).

Life, information, computation



(Conventional) Computation

Turing in '30s formalized the concept of computation and (silico-based) computer (software), von Neumann in 50s implemented the idea of computational machine (hardware).

Main ingredients:

1. **physical system** (mean to store *discrete* data, namely representing an optimization combinatorial problem)
2. *finite* number of **states**, initial state, final state
3. **program**: list of *instructions* (out of a *finite* number of predefined ones), such as reading, writing (deletion, by a blanck symbol), state changing, bidirectional movement
4. **termination criterion** to establish which (where) is the output.

What "discrete" and "finite" mean?
Any computation assumes finite and illimited space[1].

[1] Remark: limited/illimited, finite/infinite

Unconventional Computing

UC is every way to compute beyond Turing's model.

Examples: distributed, cloud, network computing (VPN: virtual private network), machine learning is part of artificial intelligence (ex. computer learning games).

Further examples: Light/optical computing (unconventional mean to store data information), DNA/RNA computing (unconventional mean and instructions), quantum computing..

The second ones are some models of *natural computing*, within a proper subset of unconventional computing. Computational models may be either inspired by nature (ex. cellular automata) or using nature to compute (ex. biomolecular computing).

Looking for new machine models

Information is carried, by a program evolution, from initial state to final one of a mean.

Computation is an **information flow** from input to output, and this may be made up of water, electricity, or something else - one of the motivations to look for new ways to perform computing is the definition of **new machine models**.

Miniaturization is another main motivation.

Richard Feynman, nobel laureate in physics (1961): “ There’s plenty of room at the bottom". He founded nanotechnology, having the key idea that molecules (and even atoms) can be machine components.

Motivation

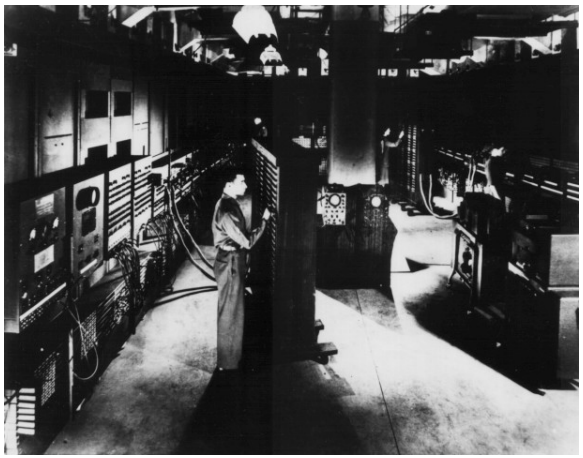
- "In the future, computers may weigh less than 1 ½ tonnes…"
- Since then, we have achieved *massive* miniaturisation of computer components

Andrew Hamilton, Brains that Click, *Popular Mechanics* **91**, no. 3, March 1949, pp. 162-167, 256, 258

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Manchester Metropolitan University

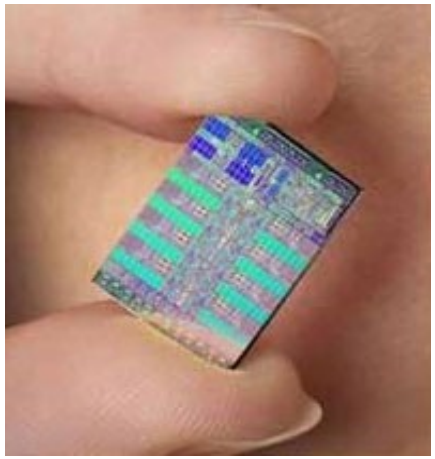
Development of computers



ENIAC (Electronic Numeric Integrator and Calculator), University of Pennsylvania, 1946

230 M^2, 3 “calculations” per second

60 years



Cell processor (PS3), Sony/Toshiba/IBM, c. 2006

25.6 gigaflops

novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

History

- Prior to World War 2, most people believed that the future lay with *analogue* computers
- Used real, continuous, *physical* quantities to represent data
- Spaghetti sort is a realisation of this idea
- Problem: constructed to solve a specific problem; difficult to generalise

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Differential Analyzer

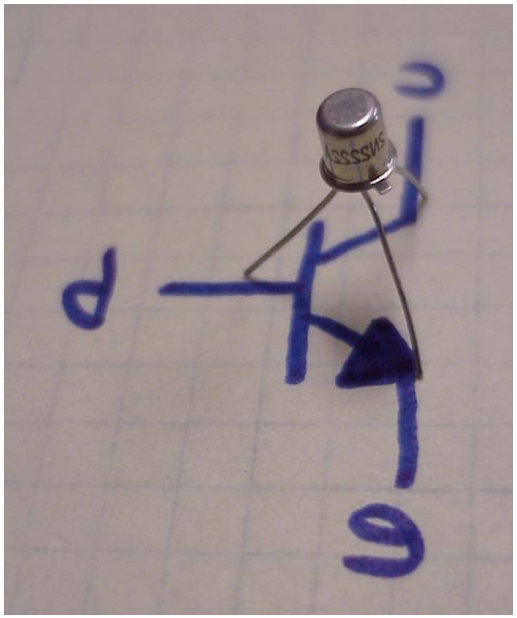
- Built by Vannevar Bush from 1927, later versions were used to design the “bouncing bomb”
- Gears, pulleys and rods to represent a set of differential equations

novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Transistor

- Fundamental building block of modern computers
- Semiconductor device, used to amplify or switch electronic signals

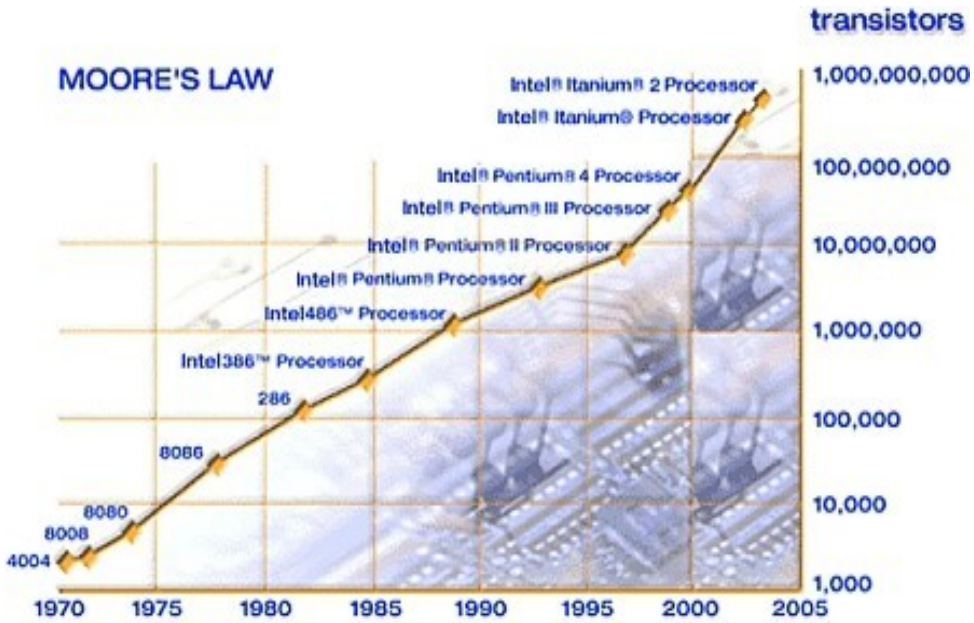


University of Arizona

novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Manchester Metropolitan University



Intel

novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Miniaturisation

- “The computing power of processors doubles every 18 months” Gordon Moore, Intel
- How true is this, now that miniaturisation is approaching atomic resolutions?
- Eventually (10-15 years?), we will hit problems as far as *traditional* fabrication techniques are concerned
- Moore's *Wall*

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Why?

- Transistors act as gates, channelling electrons around a circuit
- Their walls are thick enough to contain the electrons, and ensure that they “behave correctly”
- However, when the gates reach a critical width (c. 5nM), the electrons tunnel through the walls, causing interference

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Why?

- Also, transistor leakage manifests itself as *heat*
- "...We find that we are rapidly approaching the point where compromises are forced between device density and switching speed due to thermal constraints." George Bourianoff, Intel affiliate, 2003.
- "This looks like a fundamental limit. From a total energy point of view, you are not fooling Mother Nature." Paolo Gargini, Intel Director Of Technology Strategy

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Manchester Metropolitan University

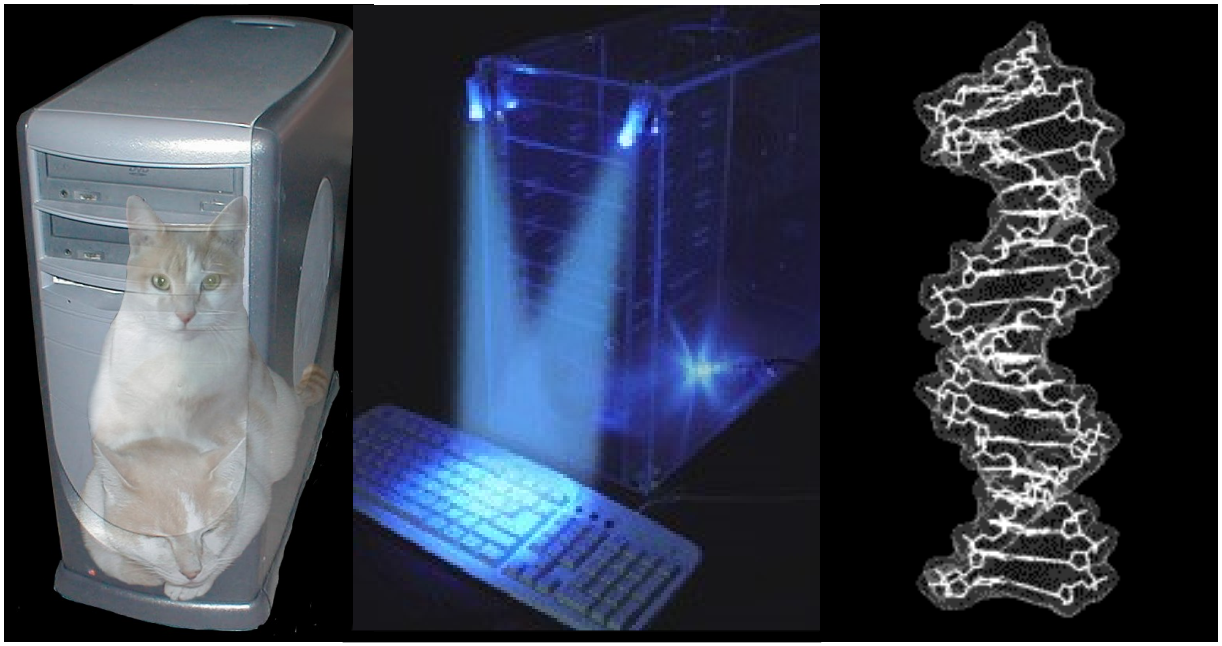
Plenty of Room at the Bottom

novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Manchester Metropolitan University

Solutions?



novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Natural Computing at different scales

Natural (complex) systems at different scales are:
Quanti $\rightsquigarrow$ Atoms $\rightsquigarrow$ Molecules $\rightsquigarrow$ Membranes $\rightsquigarrow$ Cells $\rightsquigarrow$ Organs $\rightsquigarrow$ Organisms $\rightsquigarrow$ Populations $\rightsquigarrow$ Species.

Respectively formalized in terms of computation by:
Quantum computing $\rightsquigarrow$ (molecular) self-assembly $\rightsquigarrow$ membrane/(inter-,intra-)cellular computing $\rightsquigarrow$ biological network computing $\rightsquigarrow$ population dynamics $\rightsquigarrow$ evolutionary computing

Biological computing specifies computation related to *life*, which is characterized by seven features: birth, nutrition, environmental interaction, growing, reproduction, death, evolution.

Natural Computing – a dozen trends

Algorithmic paradigms suggested by nature have typically *population-based* strategies, not anymore string evolution but data population evolution, by *massive parallelism*.

- '60 Evolutionary Computing (EC): Darwin evolution theory inspired (also combined with genetic drift)
- '60 Immunological Computing (AIS): ispired by immune system functions
- '60 Neural computing (NN, as Neural Networks): inspired by our brain functioning
- '60,'90 Amorphous Computing (AC), and molecular self-assembly: inspired by morphogenesis (ex. embryogenesis)

Natural Computing - a dozen trends

- ’85 Cellular Automata (CA): inspired to cell autoreproduction (Wolfram, a new kind of science)
- ’86 Artificial Life (AL): inspired by life features (replication, survival, resource competition, evolution)
- ’88 Swarm Computing/intelligence: collective behaviour of groups of organisms (ants, bees, insects, bacteria)

- ’94 Biomolecular (DNA, RNA, protein) computing and biomolecular machines;
- ’98 Membrane/Metabolic computing: compartments and parallel evolution by multiset rewriting rules;
- ’00 Computational genomics (project ENCODE), and synthetic biology (gene therapy, gene shuffling).

(1) Evolutionary Computing

We are given with a *fitness function* F and a *generation mechanism* G. Over an initial population, evaluate the fitness function. While F is under a threshold, *select* a subset and *expand* it by G. Stop when F exceeds the threshold, or a prefixed number of steps has been executed[2].

- **Initialize** S with a set S_0 of possible solutions
- **Evaluate** a **fitness** level F over S, and **while** F is under a given threshold, do
  - Select a subset S' of S
  - Expand S' into a population S according to G
- **Output** the population S reaching the fitness threshold

It follows population-based strategies. Darwin natural selection and genetic drift for **select**. F and threshold?

[2] see par 4.2, page 179 of the textbook (V. Manca, Infobiotics)

(2) Genetic Computing

Genetic algorithms: population of possible solutions are strings, generation is implemented by recombination and mutation (at a constant or state adapted rate). They converge "most of the times".

Memetic algorithms: solutions are *memes*, cultural entities which spead and evolve (moving in the space and being modified by the context, such as the number of individuals in that some area).

As cultural evolution versus species evolution, memetic computation is more heterogeneous and fast than genetic computation.

(3) Immunological Computing

Ingredients: B (bone marrow) cells, T (thyme) cells, and NK (natural killer) cells.
Strategy: huge recombination followed by *clonal selection* of B cells, for specific antibody-antigen attack (to kill the non-self) and *negative selection* (T cells for self-recognition).

(4) Neural Computing

Fitness function (as in evolutionary computing) to measure the performance. We are given with n neurons, m layers, form a $n \times m$ matrix, a so called *reaction function* which has to be estimated, usually by adding weights [3] on the branches.
Search of best network of (levels of) neurons, from given input and output (sort of regression). Sigmoids are good (initial) choices: $\frac{1}{1+e^{-t}}$.

[3] connection strength

(5) Amorphous Computing

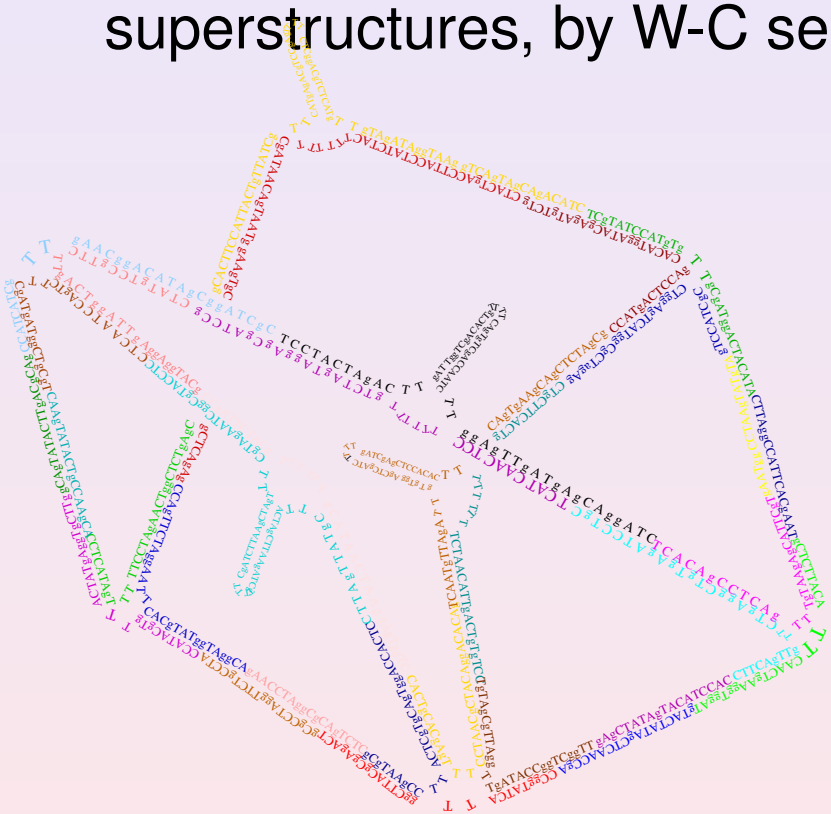
Geometrical shape generated by asynchronous molecule/cell interaction and aggregation.

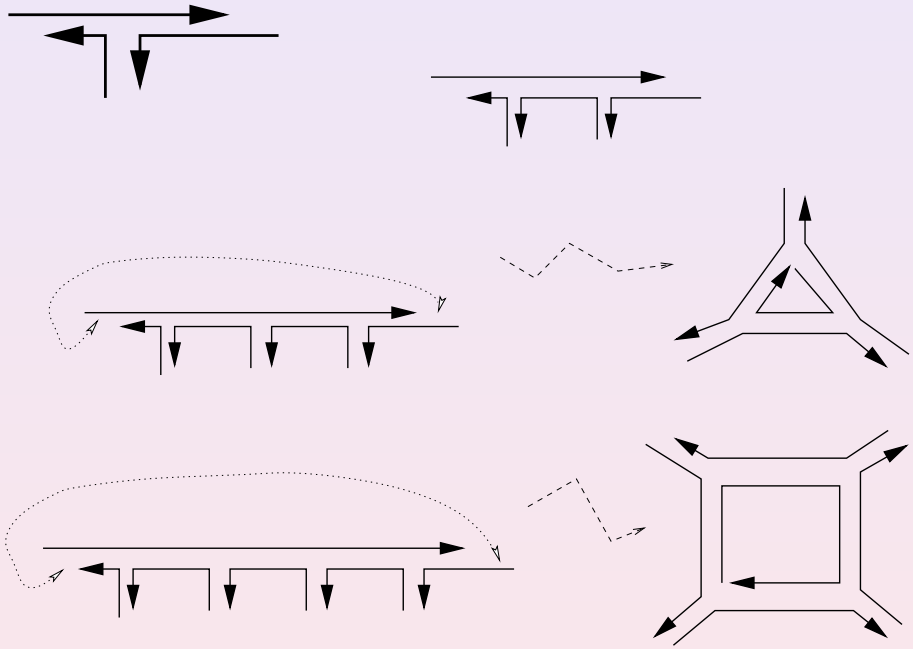
In DNA self-assembly (see following pictures), two tricks are the formation of k-branched molecules, to assembly tridimentional graphs, and DNA origami, to form any planar shape. Computation here is *effient and highly parallel*, often hard to univocally design.

Algorithms with cells aggregation capture how they split to assume new roles, and to **communicate each other (by signalling)** to form given shapes (tissues, organs, embryos).

(5) AC: Self-Assembly Algorithms

DNA substructures spontaneously self-organize into superstructures, by W-C selective affinity



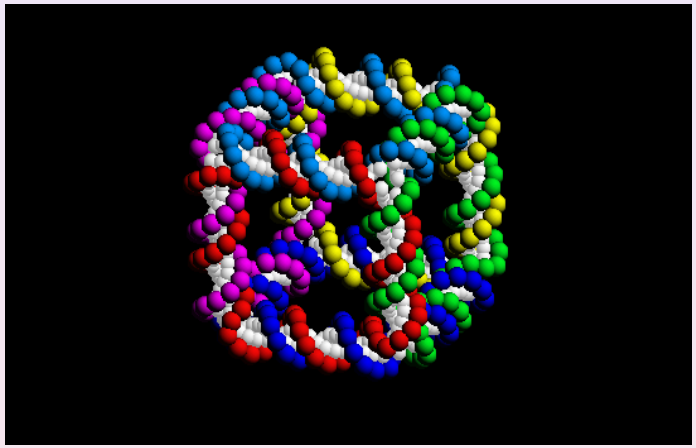


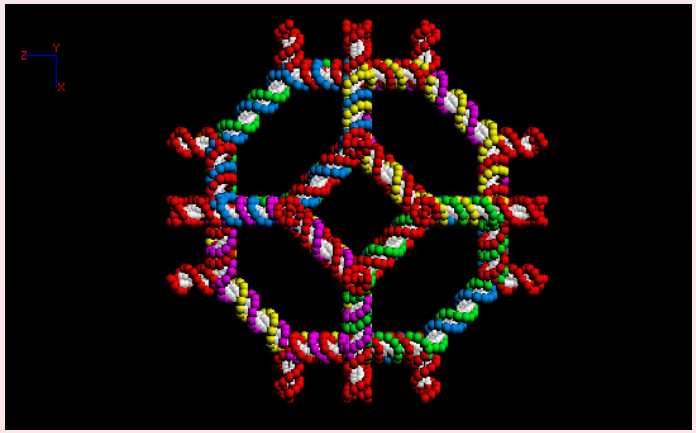
- N. Jonoska, P. Sa-Ardyen, and N.C. Seeman, *Computation by Self-Assembly of DNA Graphs*, Journal of Genetic Programming and Evolvable Machines, 4(2):123–137, 2003.

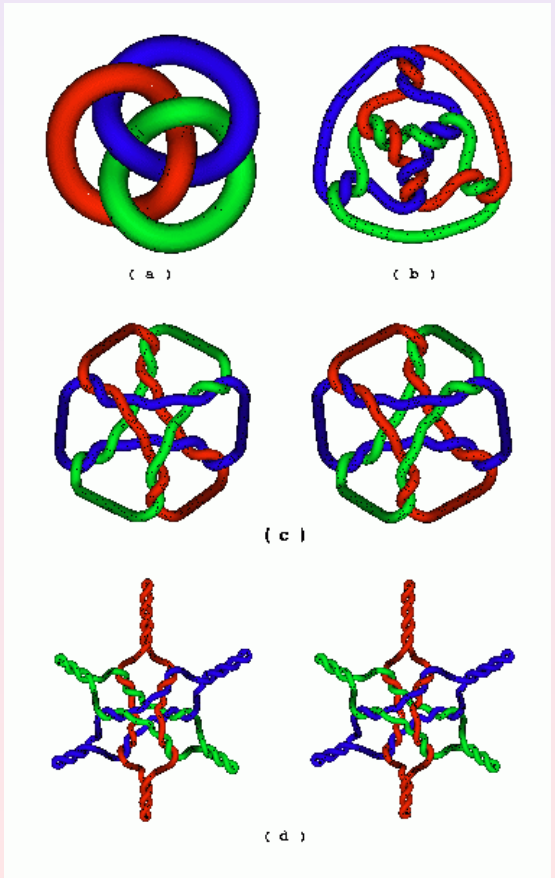
- G.F., N. Jonoska, *Forbidding-Enforcing Conditions in DNA Self-Assembly of Graphs*, Nanotechnology: science and computing (J. Chen, et. al. eds.), 105–118, 2006.

(5) AC: Self-assembly Examples

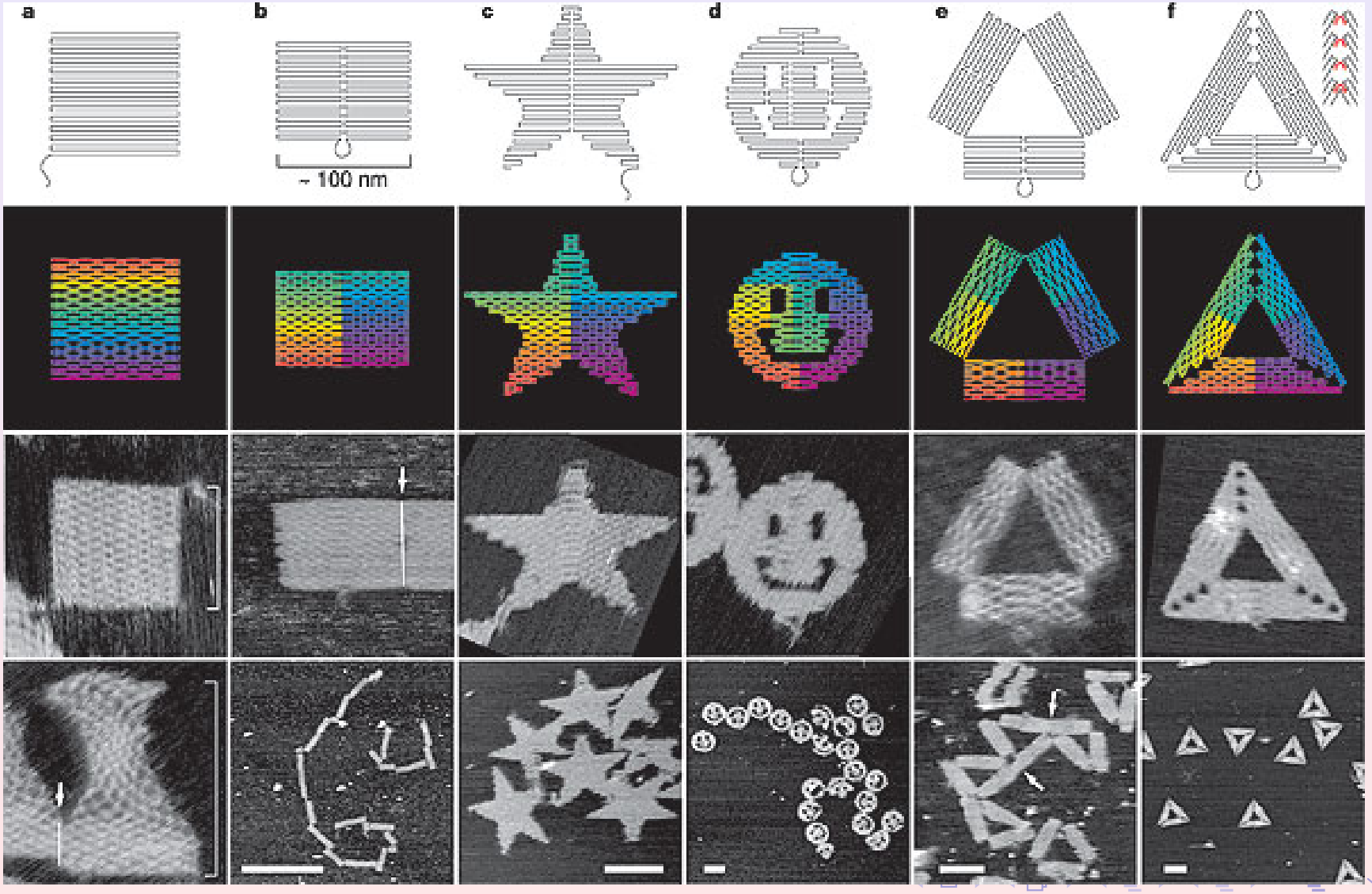
DNA cube [J.Chen, N. Seeman, Nature 1991], octahedron [Y.Zhang, N. Seeman, J.Am.Chem.Soc. 1994], Borromean rings [C.Mao, W.Sun, N. Seeman, Nature 1997], and Möbius strip (DNA14) were built up in laboratory.



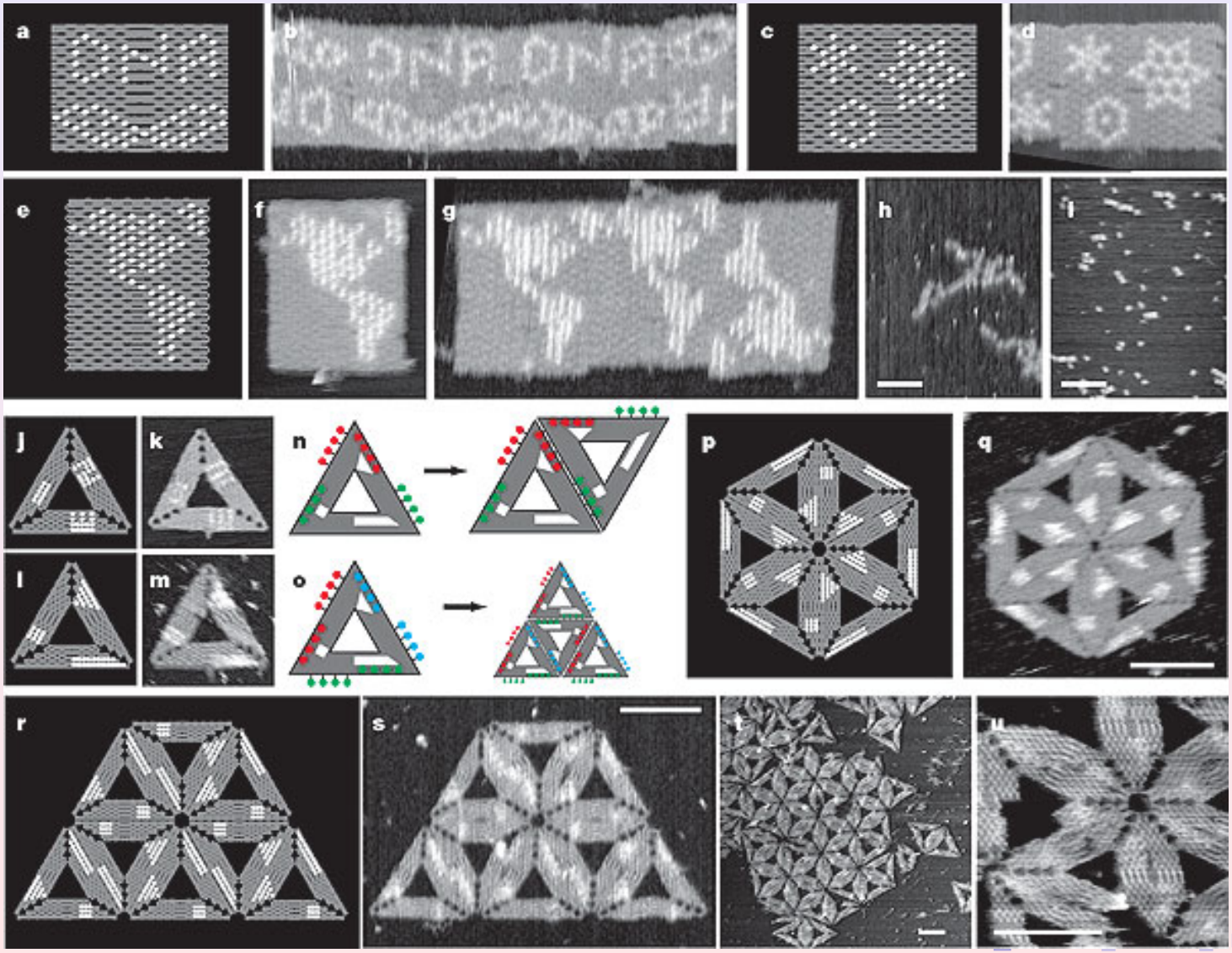




DNA Origami



DNA Origami



(5) AC – Cell communication

Engineered Communication

- Towards the end of his life, Alan Turing did some foundational work on *pattern formation* in nature, in an attempt to explain how zebras get their striped coats or leopards their spots
- The study of *morphogenesis* is concerned with how cells split to assume new roles and communicate with another to form very precise shapes, such as tissues and organs
- Turing postulated that the diffusion of chemical signals both *within* and *between* cells is the main driving force behind such complex pattern formation

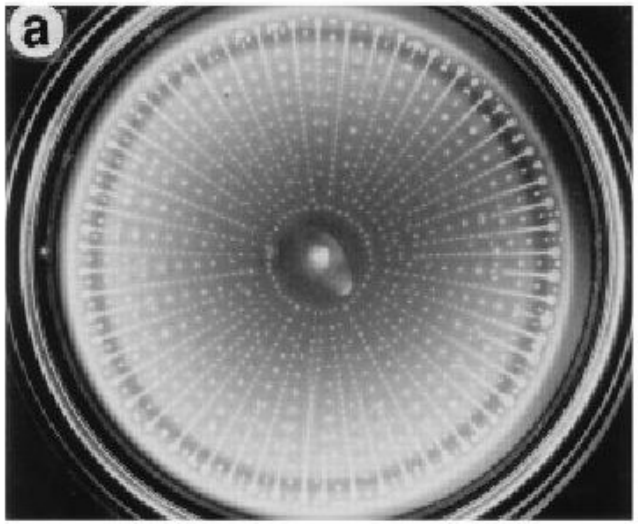
A.M. Turing. The chemical basis of morphogenesis. *Phil. Trans. Roy. Soc* B **237**:37-72, 1952

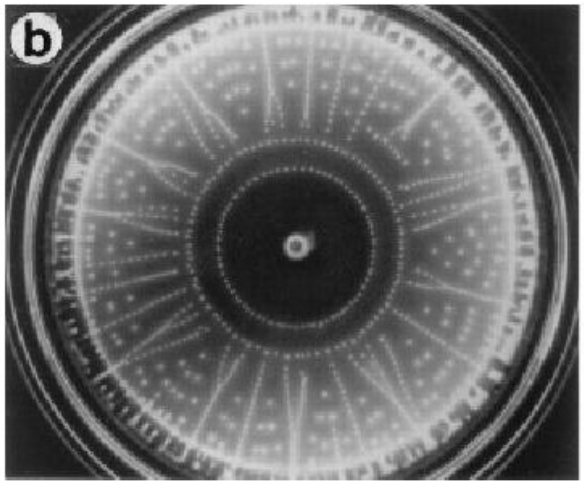
novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Manchester Metropolitan University

Pattern Formation in Bacteria





Howard Berg

novel computation group

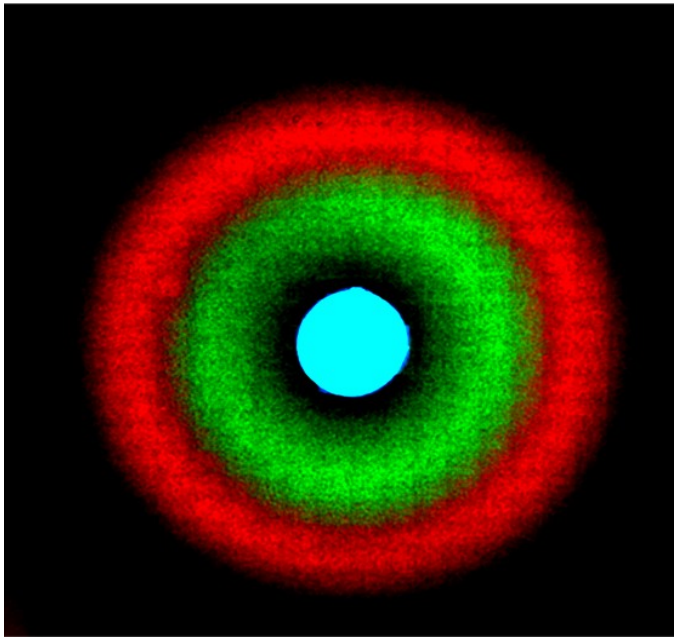
Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

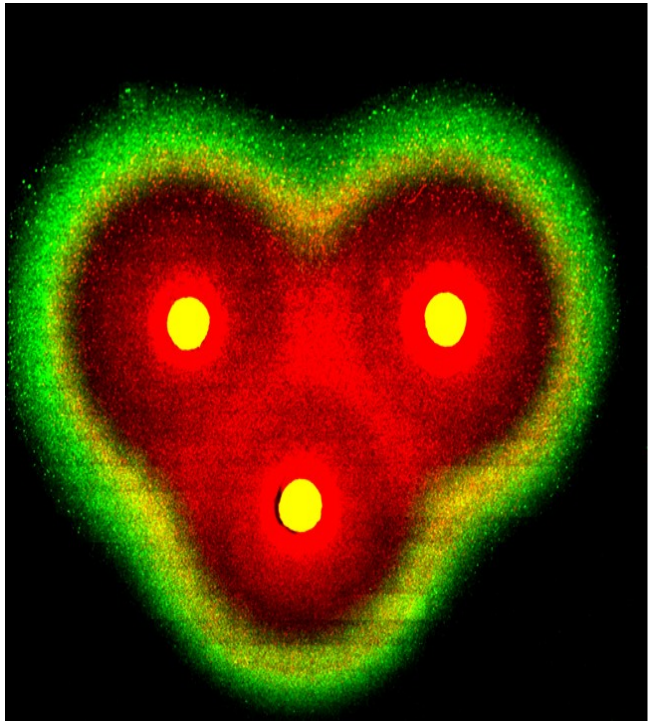
Signalling

- Although Turing's work was mainly concerned with the processes occurring amongst cells inside a developing embryo, it is clear that chemical signalling also goes on between *bacteria*
- Ron Weiss (Princeton) particularly interested in Vibrio fischeri, a bacterium that has a symbiotic relationship with a variety of acquatic creatures, including the Hawaiian squid
- This relationship is due mainly to the fact that the bacteria exhibit *bioluminescence* – they generate Luciferase (coded by the *Lux* gene), a version of which is also found in fireflies, and which causes them to glow when gathered together in numbers

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Manchester Metropolitan University





Ron Weiss

novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

(6) Swarm computing

Intelligence: sophisticated patterns emerge as the population behaviour of simple agents, interacting by means of local rules. Ex. ACO = ant colony optimization, with pherormone-based search for "geodedica" (traffic and crowd on shortest paths). Population may have an internal structure, different roles, partial solutions.

No coordination, but individual behavious driven by common finalities. Multiagent systems acting according to population-based cooperative algorithms which implement: try, compare, modify, *communicate*. They may have competition.

Information sharing may happen by a network with some scoring – agents update their current state better than the others. When a level of satisfaction is reached, agent publishes the solution (its state) and the computation terminates.

(7) Cellular Automata (CA) – oneD

Time is discrete, and *current t*-state is computed by previous *(t-1)*-states of neighbours. All cells have a same updating rule.

Ex. “the simplest non-trivial cellular automaton” (Wolfram, ’84) generates pseudo-random integers for Mathematica.

Evolution is not periodic if we apply rule 90 (exclusive OR) to a binary automaton of dim n, with random initial state (with 1 at $\frac{n}{2}$ position, or finitely many nonzero cells, the computation is a “fractal” replicator).

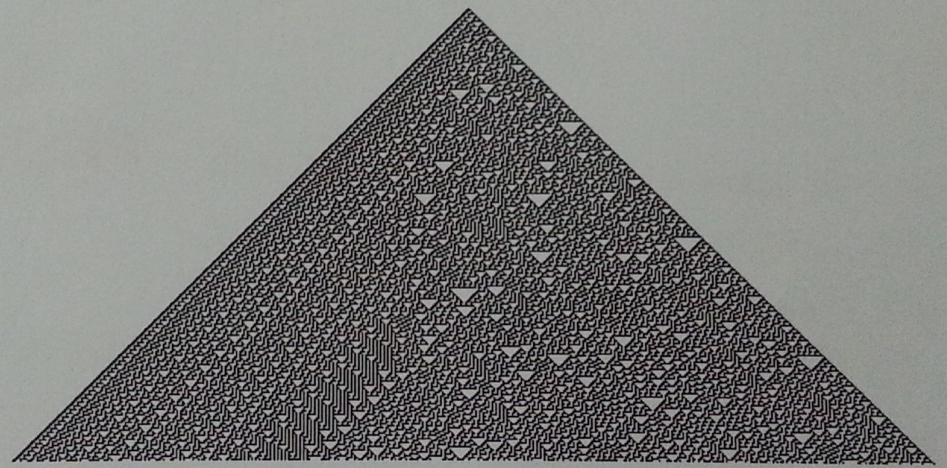
Sierpiński triangle

Cellular Automata (2)

Example:

111	110	101	100	011	010	001	000
0	0	0	1	1	1	1	0

Starting from an array of size N with 1 only in position $N/2$:



The evolution seems to be non-periodic

- This technique is used in the tool Mathematica to generate pseudo-random integers

(7) Cellular automata (CA) – twoD

Grid of cells (on Z^2, concept of distance), with a finite number of states and a function with a scope limited to the adjacent cells. In other words, a grid of tabular functions.

Ex. One cell is alive (value 1) if (at least) three of its adjacent cells are alive. The system is monotone, if once a cell dies (value 0) it never becomes alive again (back to value 1).

Game of Life is a CA in 2D proved to be universal.

(8) Membrane computing (Gh. Păun, '98)

We are focused on observing nature, and reproducing it, not to solve any biological problem. This part is called bio-inspired computation [4]. A notable example: membrane computing [5], with a literature long almost 20 years [6].

Def: $M = (A, \mu, R_1, R_2, \ldots R_m, w_1, \ldots, w_m)$, wih A alphabet, $\mu \in \mathbb{N} \times \mathbb{N}$ membrane structure of degree m, R_i set of rewriting rules and w_i multiset of objects from A, for $i = 1, \ldots, m$.

Rewriting rules on commutative strings applied by maximal parallelism (prob., priority, determinism).

[4] par 4.2, page 179 textbook
[5] par 2.7, page 75 textbook
[6] `http://ppage.psystems.eu`

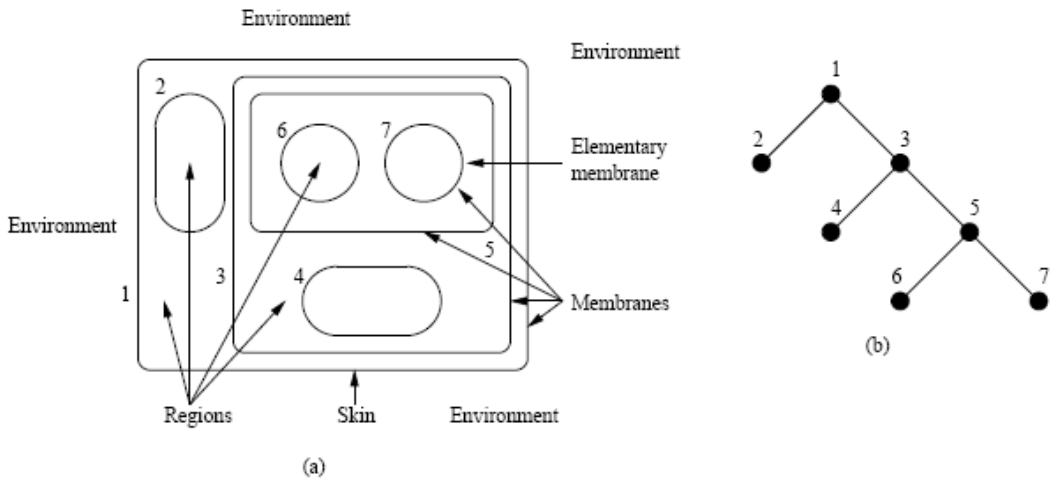
Diagrams and trees

Membranes as separation, selection, concentration protection.

Manchester Metropolitan University

We attach an integer label to each membrane in order to make it addressable

Since each region is delimited by a unique membrane, we use membrane labels to label regions

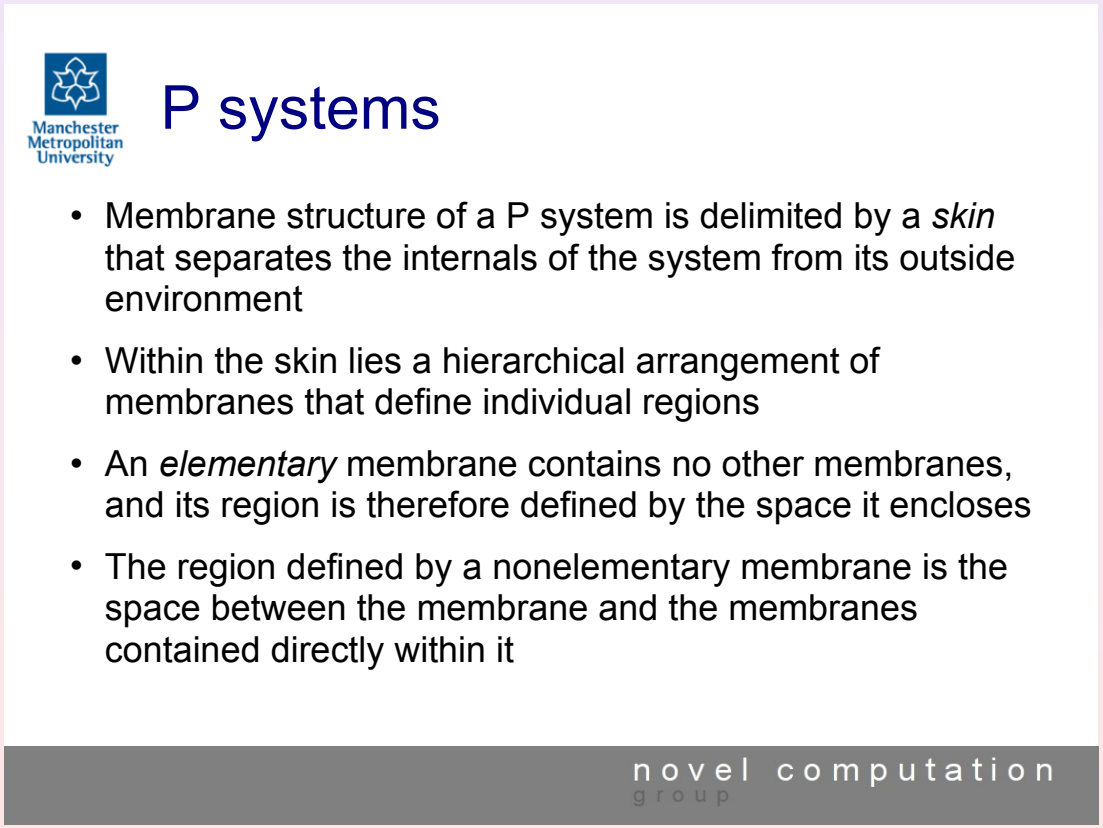


novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Membrane structure

Membrane structure employed both to model and to compute (duality between membranes and objects - multisets of membranes) - strings, trees, diagrams, binary relation.



Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Membrane Rules

P systems

- Each region contains a multiset of *objects* and a set of *rules*
- Objects are represented by symbols from a given alphabet
- Rules transform or "evolve" objects, and are of the form *before* $\rightarrow$*after*, meaning "evolve every instance of *before* into an instance of *after*"
- As we are considering multisets, rules may be applied to multiple objects

novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Maximal parallelism (and non-determinism)

- Parallelism: More than one rule may be applied (on different objects) in the same step
- Multiset: each rule may be applied more than once in the same step (on different objects)
- Maximality: A multiset is chosen non-deterministically such that no other rule can be applied to the system obtained by removing all the objects involved by that choice.

Example: Given $\begin{cases} c \to a \\ a \to b \\ a \to bc \\ b \to bc \\ b \to ab \end{cases}$ of rules r_1, r_2, r_3, r_4, r_5,
and starting from multiset *abccab*: how many multisets you obtain by applycation of the above rules by non-det. max. par?

(9) Metabolic computing - an example

Rules above for a metabolic system, where M represents the initial state $X(0)$. Compute $X(1)$, according to the flux maps: $u_1 = f_1(a, b, c) = ab$, $u_2 = f_2(a, b, c) = c^2$, $u_3 = f_3(a, b, c) = 2a$, $u_4 = f_4(a, b, c) = a$, $u_5 = f_5(a, b, c) = c$.

By EMA: $X[1] = A \times U[0] + X[0]$, where:

$$A = \begin{pmatrix} 1 & -1 & -1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \end{pmatrix}, U[0] = \begin{pmatrix} 4 \\ 4 \\ 4 \\ 2 \\ 2 \end{pmatrix}, \text{ and}$$

$$X(0) = \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix}, \text{ therefore } X(1) = \begin{pmatrix} 0 \\ 10 \\ 4 \end{pmatrix}.$$

(10) DNA computing

$$\text{Input data} \overset{encoding}{\longleftrightarrow} \text{DNA} \qquad \overset{bio\text{-}steps}{\rightsquigarrow} \qquad \text{DNA} \overset{decoding}{\longleftrightarrow} \text{Output data}$$

Information is stored in biopolymers, enzymes (mol bio, gen eng techns) manipulate them in a massively parallel way, according to strategies producing universal computation

- G.F., M. Margenstern, *A DNA computing inspired computational model*, TCS 404: 88–96, 2008.

Operations designed for *multisets of strings* over $\{A, T, G, C\}$ are performed over molecules in a test tube. *Bioalgorithm:* sequence of operations s.t. final molecules encode the solution (i.e., final *DNA pool* contains the solution).

DNA Algorithms: indetermination and determinism

Bio-algorithms driving system behaviours can be written as a sequence of operations, such that the final pool of molecules contains the desired solution.

Indetermination. A feature common to molecular and quantum computations is the possibility to be *observed and captured only at certain moments*. In both cases, the step by step exact effect of the operations cannot be controlled, and the system is altered by observing the result.

Determinism. DNA algorithms transform macro-states of DNA pools, where the *population properties* are controlled by the macro-steps while the *individual non-determinism* is left unaffected [V. Manca, G.F., Mathematical Bioscience 211, 282-298, 2008].

Importance of Experimentation

Title of Robin Milner's lecture, inauguration of the Lab. for Foundation of Computer Science at the Dept of Computer Science, Univ. of Edinburgh (1986): Is Computing an Experimental Science?

Laboratory experiments are crucial for DNA computing: the algorithms *need* to be validated by the comparison with experimental results.

"No amount of experimentation can ever prove me right; a single experiment can prove me wrong." (A. Einstein)

Lab suggestions: looking for right level of abstraction (encoding, mismatches, melting temperatures) considering the efficiency, the feasibility, and the lasting of operations, the complexity in space, the cost both in time and money.

Other trends in DNA computing

DNA computing as a new computational model to generate formal languages (Turing-completeness of H-systems – languages generated by splicing).

Computational genomics to understand how information is organized/structured into genomes, given the way we observe it is transmitted. Alignment-based algorithms and alignment-free (dictionary-based) methods. Information theory and formal languages are combined to study *very long* sequences with regularity properties to discover mainly by observed data.

DNA computing to design new biotechnologies (ex. algorithms for super-gene extraction or DNA libraries generation). Major quantity of protein (insulin) by inserting some part of human genome to the bacterial one, different (artificial) protein by exchanging the gene order (industrial or medical purposes).

(11) Computational genomics

Study of genomes is referred to as **genomics** - while **genetics** concerning the study of (single of groups of) genes.

Goal: Inference of sequence homology and function. Difficoult problems: Definition of good similarity/distance functions, and development of efficient algorithms for their computation.

Local sequence (similarity) analysis is traditionally performed by **alignment-based methods** ([V. Brendel et al. '84], FASTA, BLAST). Recent **alignment-free methods** [S. Vinga et al. '03] in computational genomics focus on a systemic view, based on empirical studies of frequencies of DNA k-mers (genome factors of length k) in whole genomes [Y. Fofanov et al., 04, B. Chor et al. '09, D.B. Searls '10].

Alignment based Methods

Global methods align, for example: axab - cs with ax-bacs.
Local methods align, for example: pqr**axabcs**stvq and yx**axbacs**pq. In both cases, similarity of two strings or parts of them, is measured.

Limits: All algorithms (for proteins and DNA) need a scoring value: PAM, BLOSUM, gap penalities..
No shuffling recombination or substring interchange operation allowed, performance does not scale on a genome-wide study.

Alignment-free methods

Intuition: String similarity is based only on the dictionary of substrings that apper in the two strings, irrespective of their relative position. Two strings are similar if they are composed of the same basic building blocks, ex: **abra**cad**abra**, **abra**ca**abra**d, or share many building blocks (which length?).

No parameter setting, no training, no learning. This approach favors speed and scalability of computation (usually time linear in the size of input).

Powerful filtering tools to screen a data set for similarity. Broader spectrum of applications: genome-wide comparison across species, identification of (Cis Regulatory) modules with a common function, of coding regions, of tandem repeats, protein classification.

Infogenomics: basic idea

Looking at a *very long* string over an alphabet of four symbols as a text, which language has to be deciphred. [7] Information is comprised in words, initially broken down as a fixed-length code.

Sequences become bags of words, any investigation and comparison (e.g., similarity analysis) is made on dictionaries.

Explicit collection and use of word statistics, similarity/distance of points in high-dimensional geometric spaces (e.g., 4096), common substring counts, informational indexes

[7] A. Castellini, G. Franco, V. Manca. BMC Genomics 13(1), 485 (2012).

Information theory

Information theory deals with efficient and effective communication of data: compression and transmission processes, statistical encoding of data and signals, error finding in received messages (auto-correcting code).

Infogenomics employs methods from FLT and informational approaches (*dictionaries, codes*) to define genomic indexes, sequence similarity measures, gene networks.

A code is a function from a set $T^{\star}$ (strings over $T = \{0, 1\}$, for example) into a finite set S of data.
Es. Genetic code (codons $\rightarrow$ aminoacids, T set of nucleotides).

Genetic code

First letter	Second letter: U	Second letter: C	Second letter: A	Second letter: G	Third letter
U	UUU, UUC: Phenylalanine	UCU, UCC, UCA, UCG: Serine	UAU, UAC: Tyrosine	UGU, UGC: Cysteine	U C A G
	UUA, UUG: Leucine		UAA, UAG: Stop codon	UGA: Stop codon; UGG: Tryptophan	
C	CUU, CUC, CUA, CUG: Leucine	CCU, CCC, CCA, CCG: Proline	CAU, CAC: Histidine; CAA, CAG: Glutamine	CGU, CGC, CGA, CGG: Arginine	U C A G
A	AUU, AUC, AUA: Isoleucine	ACU, ACC, ACA, ACG: Threonine	AAU, AAC: Asparagine; AAA, AAG: Lysine	AGU, AGC: Serine; AGA, AGG: Arginine	U C A G
	AUG: Methionine; start codon				
G	GUU, GUC, GUA, GUG: Valine	GCU, GCC, GCA, GCG: Alanine	GAU, GAC: Aspartic acid; GAA, GAG: Glutamic acid	GGU, GGC, GGA, GGG: Glycine	U C A G

ATTCACAGTGGA

I H S G

Slide courtesy of prof. Bin Ma, University of Waterloo, CA

Motivation, open problems for Infogenomics

◇ How information is structured within genomes - how genes communicate each other (and with their distal promoters) in the transcriptional process?

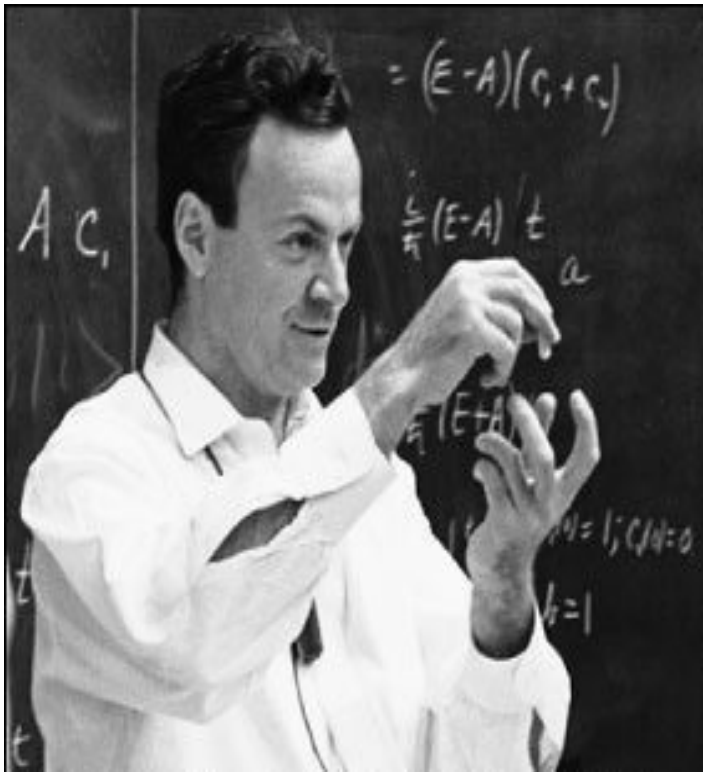
◇ Regularity properties, informational indexes characterizing (classes of) genomes

◇ Genome discrimination methods, namely for phylogenetic or medical purposes

◇ A model which explains how major informational processes work to keep the cell alive, by means of an interplay among its metabolism, growth and duplication.

Manchester Metropolitan University

Synthetic Biology



"What I cannot create,
I cannot understand"

– Richard P. Feynman

novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

(12) Synthetic Biology

Definitions

- An emerging field at the intersection of biology, computer science, chemistry, physics, mathematics, ... that aims to design and build novel biological systems (and therefore understand them better)
- Eventually, we would like to do this in the same way as engineers design and build mechanical or electronic systems
- Hierarchical and modular features allow biological systems to be understood, and make SB possible

novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

(12) Synthetic Biology

Why use Cells?

- Unique features:
  - small, self-replicating, energy-efficient
- Purposes:
  - Biomedical applications
  - Environmental applications (sensors & effectors)
  - Embedded systems
  - Interface to chemical world
  - Molecular scale engineering

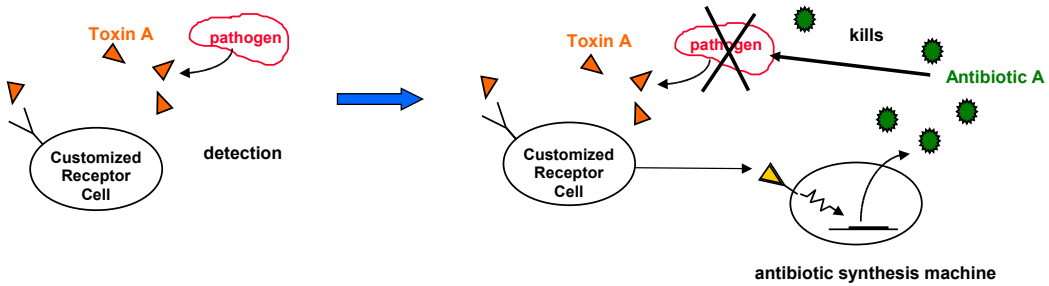
novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

(12) Synthetic Biology

Microbial Robotics

- Potential to engineer behavior into bacterial cells:
  - phototropic or magnetotropic response
  - control of flagellar motors
  - chemical sensing and engineered enzymatic release
  - selective protein expression
  - molecular scale fabrication
  - collective behavior
    - autoinducers
    - slime molds
    - pattern formation
- Example: timed drug-delivery in response to toxins



Ron Weiss

novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

(12) Synthetic Biology

Genetic Process Engineering - Dry

- Methodology for modifying the DNA encoding of existing genetic elements to achieve desired input/output behaviour for constructing reliable circuits of significant complexity
- Construct library of well-characterised (understood) genes, with their inputs and outputs defined – circuit components
- Take a circuit for a given task (eg. if-then-else clause) and map it onto the component library

novel computation group

Slide courtesy of prof. Martin Amos, Manchester Metropolitan University, UK

Progress towards miniaturization

Natural systems at different scales:

Quanti $\rightsquigarrow$ Atoms $\rightsquigarrow$ Molecules $\rightsquigarrow$ Membranes $\rightsquigarrow$ Cells $\rightsquigarrow$ Organs $\rightsquigarrow$ Organisms $\rightsquigarrow$ Populations $\rightsquigarrow$ Species.

have been investigated by a computational prospective in a chronological order inverse to the scale:

- '60-'80 (EC, ACO, NN, CA),

nano: '90-'00 (Membrane and biomolecular computing),

- more recently, optical (foton-based) computing and QC: information elaboration by principles of quantistic mechanics, first actual computer in 2001 (IBM), first in commerce (D-wave One) in 2011, first available on cloud Feb 2016 (`http://www.research.ibm.com/quantum/`).

Looking at nature with new glasses

The *evolutionary process* by which proteins and genetic sequences are designed in nature involves recursive cycles of *diversification* (by recombination and/or mutagenesis), followed by *differential amplification* based on the fitness of each sequence in a complex environment.

Natural computation provides a real-time and real-environment computation having as outcome *a sequence or a structure* rather than a numerical solution. Dynamics/computation/life is often more important than output (system death).

Looking at nature with new glasses

It occurs at a variety of scales, depending on whether the selectable unit is a gene, a whole organism, or a comunity of organisms. The basic process is the same: involving *self-computation* of each selectable unit's *fitness*, and thereby the composition of the entire *population*, based on diverse and repeated *interactions with the local environment*.

Improved understanding of the natural design processes has greatly helped us develop better methods of designing proteins, whole genomes, as well as methods of numerical molecular computation.