Genetic Algorithms and Genetic Programming

Lecture 17: (24/11/09)

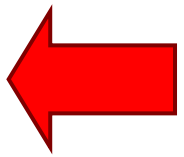
DNA computing and Membrane computing

Michael Herrmann

michael.herrmann@ed.ac.uk, phone: 0131 6 517177, Informatics Forum 1.42

Overview

I. GA (1-7)
II. GP (8-10)
III. ACO (11-13): Ant colony optimization
IV. PSO (14-16): Particle swarm optimization
V. Differential evolution, Metaheuristic search (16)
VI. NC (17): Overview on DNA computing, Membrane computing

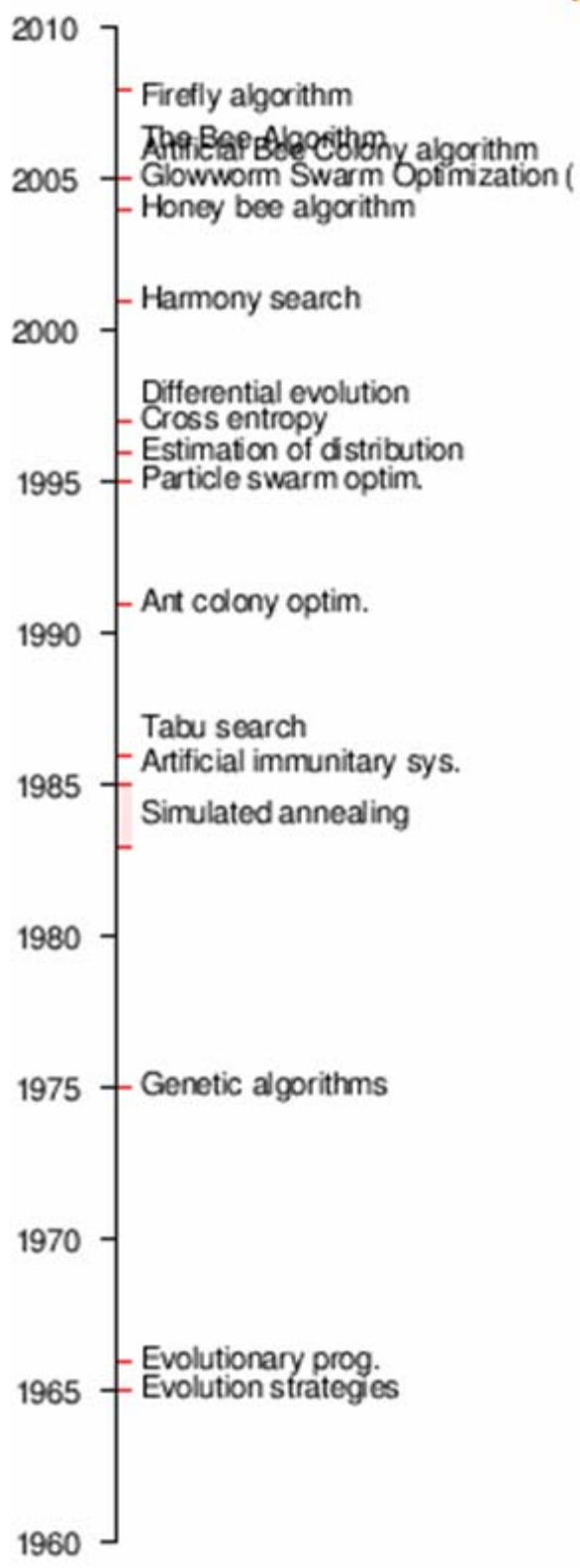


Not included:

artificial neural networks, quantum computing, cellular automata, artificial immune systems, amorphous computing, molecular computing, organic computing

Meta-Heuristic Search

- μετα “beyond”, ευρισκειν "to find“
- applied mainly to combinatorial optimization
- The user has to modify the algorithm to a greater or lesser extend in order to adapt it to specific problem
- These algorithms seem to defy the no-free lunch (NFL) theorem due to the combination of
  - biased choice of problems
  - user-generated modifications
- Can often be outperformed by a problem-dependent heuristic



Natural computing

- Complex dynamics of a bio-physical system
- Parallel by nature
- Non-deterministic

- Encoding problems must be solved in applications to practical problems
- Computation may be ineffective or inefficient on some problems

We turn to biology not just as a metaphor, but as an actual implementation technology …

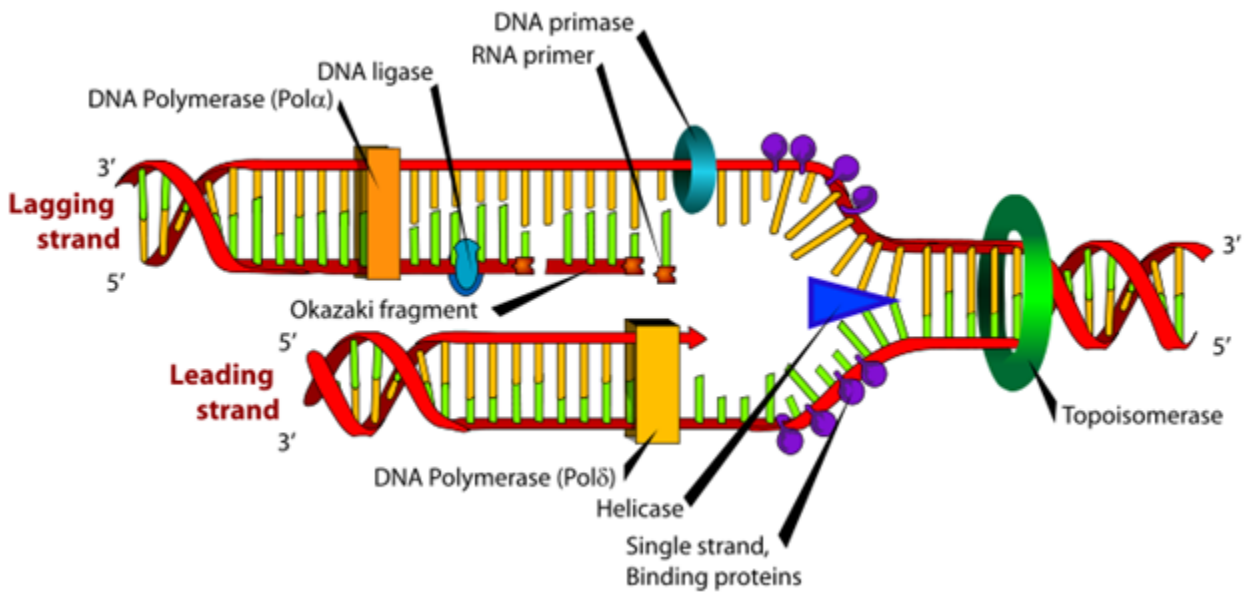
Harold Abelson et al., MIT, 2000

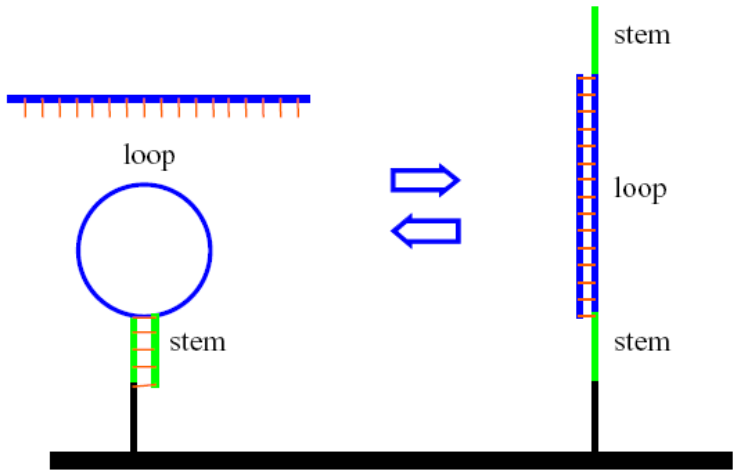
DNA Computing in Numbers

- More generally: molecular computing
- History
  - 7-point Hamiltonian path problem (L. Adleman 1994)
  - Programmable molecular computing machine (2002)
  - DNA computer (2004)
- Energy consumption $2*10^{19}$ op/Joule (a billion times better than classical computers)
- 5 grams of DNA contain 10^{21} bases (Zetta Bytes)
- Each DNA strand represents a processor (300 trillion)
- Speed: 500-5000 base pairs a second.
- Still the fastest and most economical of the existing computing mechanisms

DNA Computing

- **Parallelism** by (typically) trillions of “processors”
- **Complementarity** makes DNA unique (error correction)
- **Basic suite of operations:** AND,OR,NOT in a CPU must be represented by cutting, linking, pasting, amplifying or repairing DNA





cellbiology.med.unsw.edu.au

Liu et al. BioSystems 77 (2004) 87–92

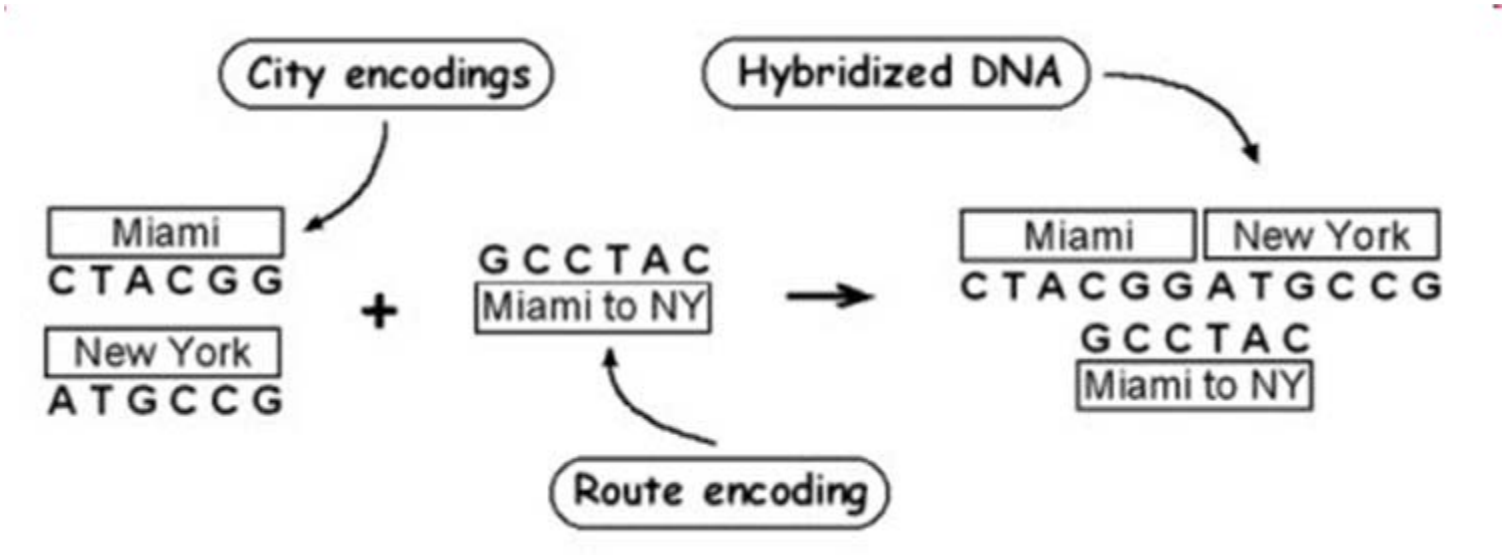
Adleman's solution of the Hamiltonian Directed Path Problem (HDPP).

1. Generate random paths
2. From all paths created in step 1, keep only those that start at s and end at t.
3. From all remaining paths, keep only those that visit exactly n vertices.
4. From all remaining paths, keep only those that visit each vertex at least once.
5. If any path remains, return "yes"
   otherwise, return "no"

Example with 5 cities: Step 1

Will Ryu at
Ars technica

city	code
Los Angeles	GCTACG
Chicago	CTAGTA
Dallas	TCGTAC
Miami	CTACGG
New York	ATGCCG



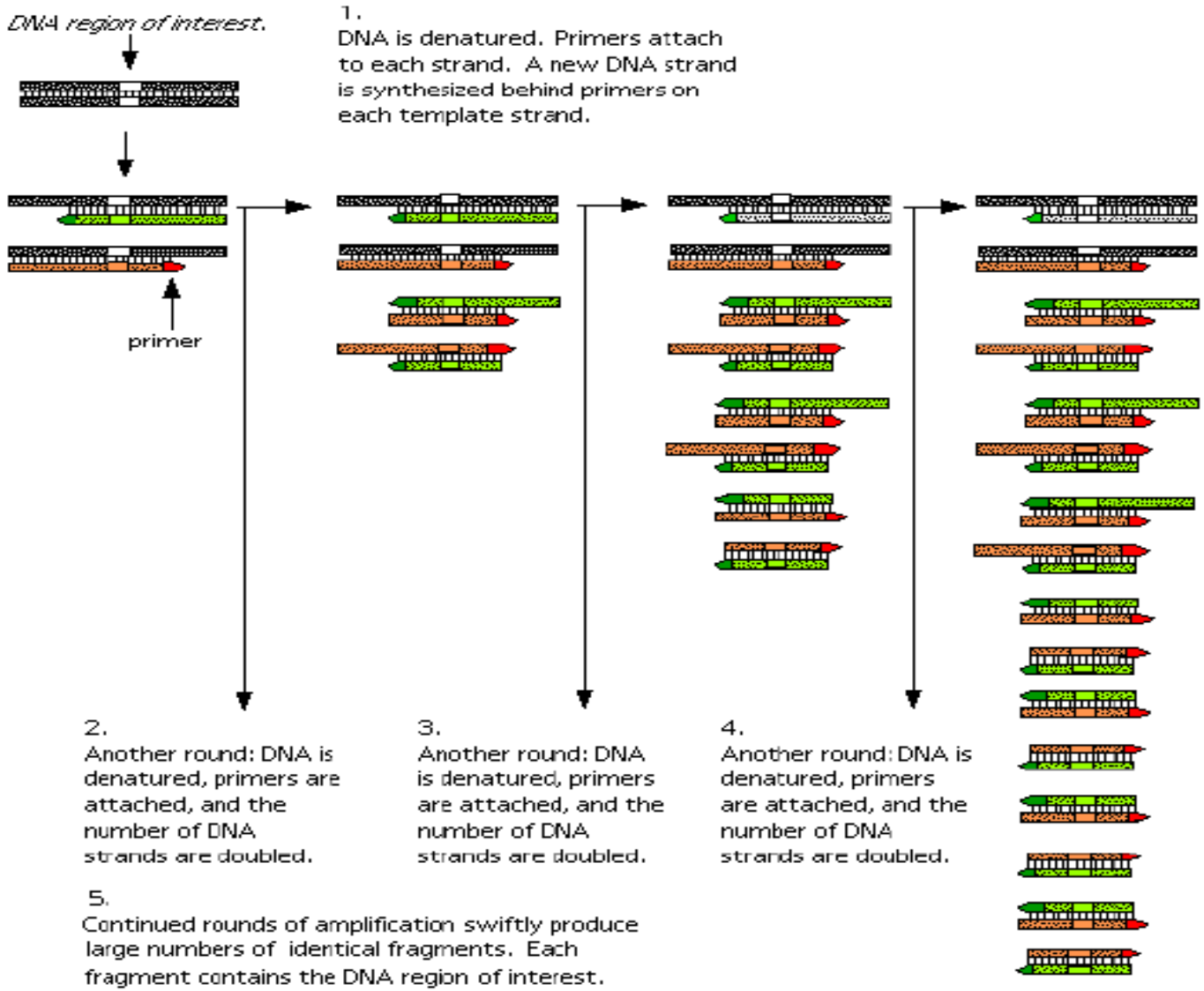
L.A -> Chicago -> Dallas -> Miami -> New York would simply be GCTACGCTAGTATCGTACCTACGGATGCCG

Polymerase Chain Reaction (PCR)

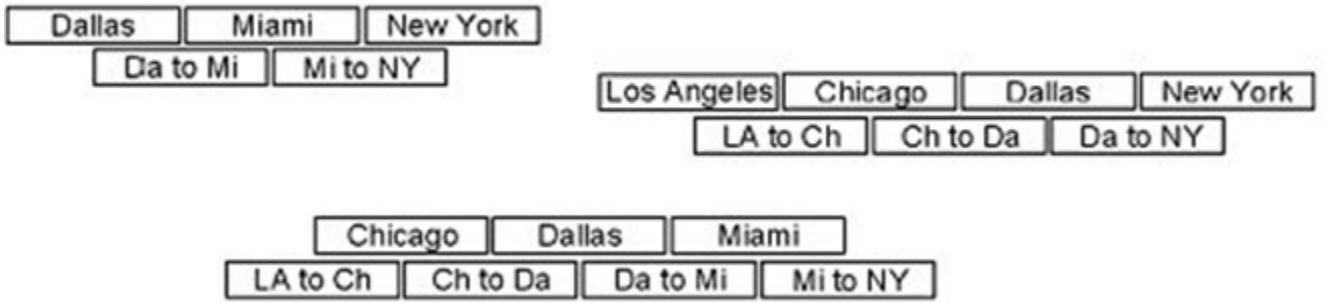
PCR: One way to amplify DNA.
(Kary Mullis, 1985)

PCR alternates between two phases: separate DNA into single strands using heat; convert into double strands using *primer* and polymerase reaction.

PCR rapidly amplifies a single DNA molecule into billions of molecules



Part 2: PCR

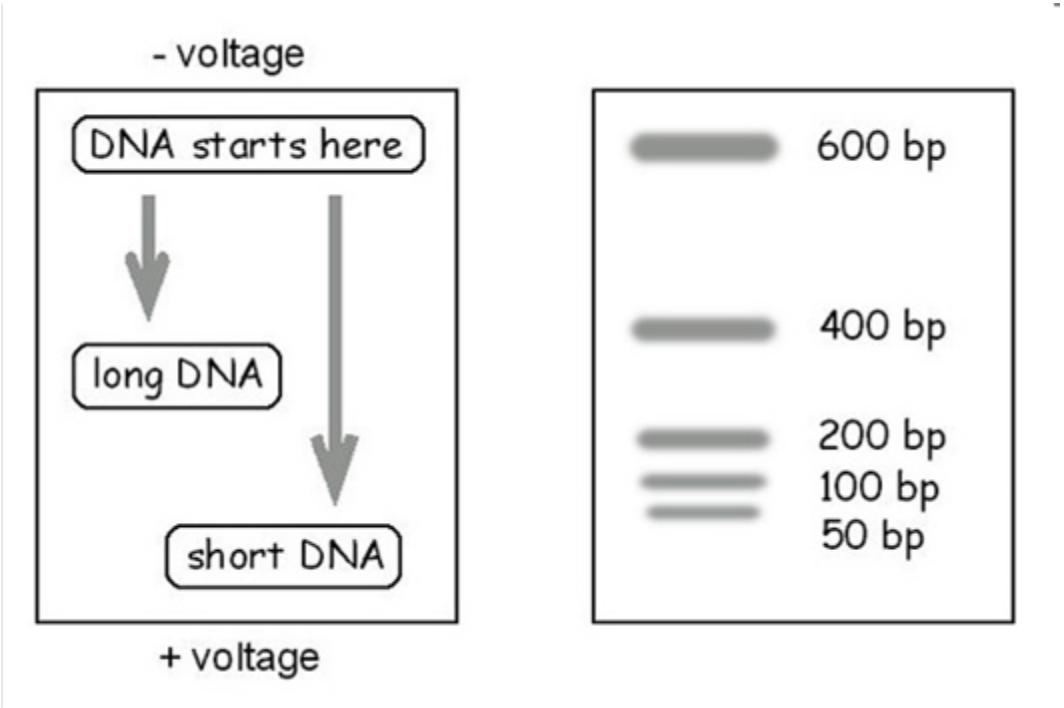


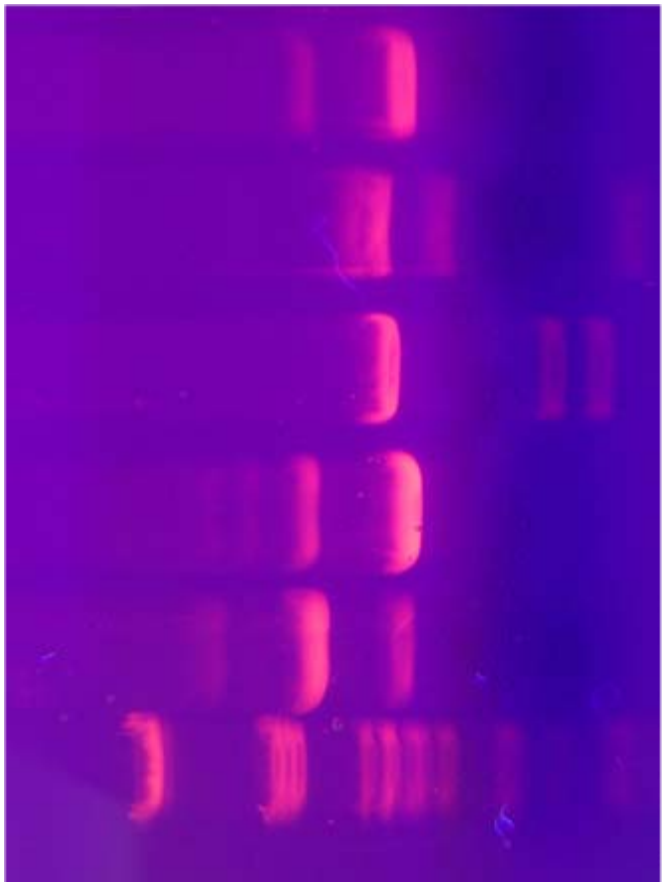
(a billion copies of each)

- Starting a PCR with the starting city as a primer (i.e. the primer is the complement of the city code)
- and a primer for termination corresponding to the goal city.
- produces lots of strands with the correct start and end (but with variable lengths and with possible repetitions)

Part 3: Gel electrophoresis

- Used to measure the length of a DNA molecule.
- Smaller DNA molecules are travel faster in an electric field (for same charge)





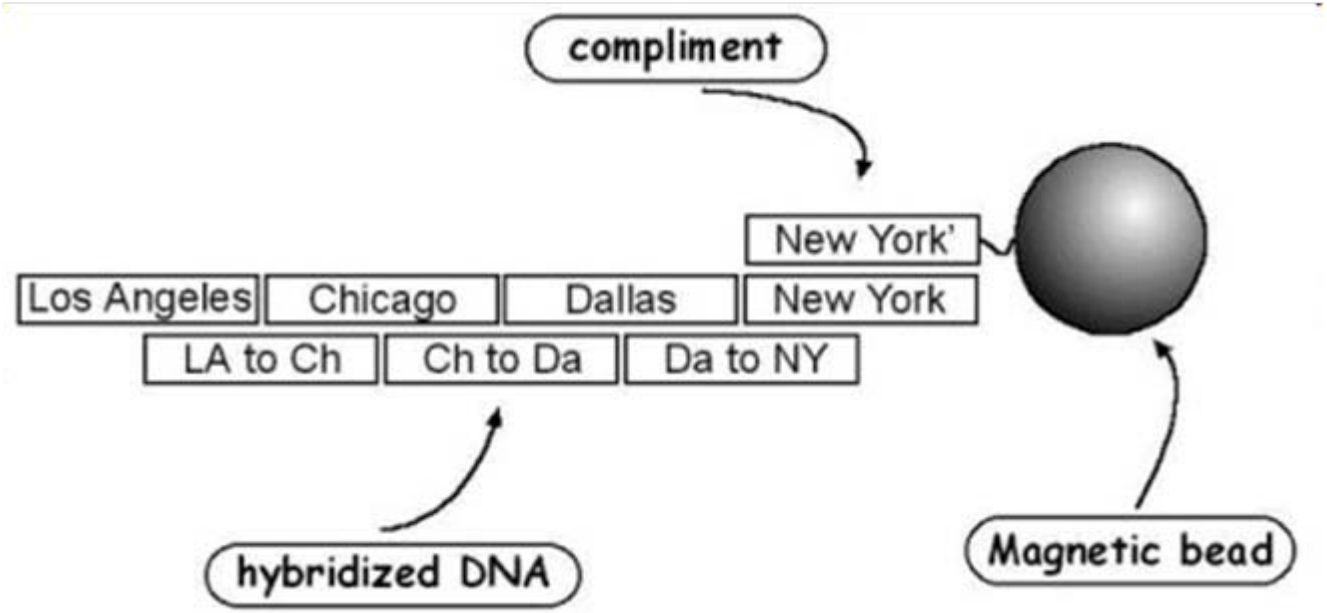
http://arstechnica.com/reviews/2q00/dna http://de.wikipedia.org/wiki/Gelelektrophorese

Part 3: Gel electrophoresis

- From all remaining paths, keep only those that visit exactly n vertices.
- Isolate the DNA that was 30 base pairs long (5 cities times 6 base pairs).

Step 4: Affinity purification

- Select itineraries that have a complete set of cities
- Sequentially affinity-purify fives times, using a different city complement for each run.



- We are left with itineraries that start in LA, visit each city once, and end in NY.

http://arstechnica.com/reviews/2q00/dna

Step 5: Reading out the answer

- Result can be obtained by sequencing the DNA strands
- More effectively by using **graduated PCR:**
  - A series of PCR amplifications using the different primer for each city in succession.
  - Measuring the various lengths of DNA for each PCR product reveals the final sequence of cities.
  - For example, starting with LA gives 30 base pairs. If LA and Dallas primers give 24 base pair, Dallas is the fourth city in the itinerary.

Caveats

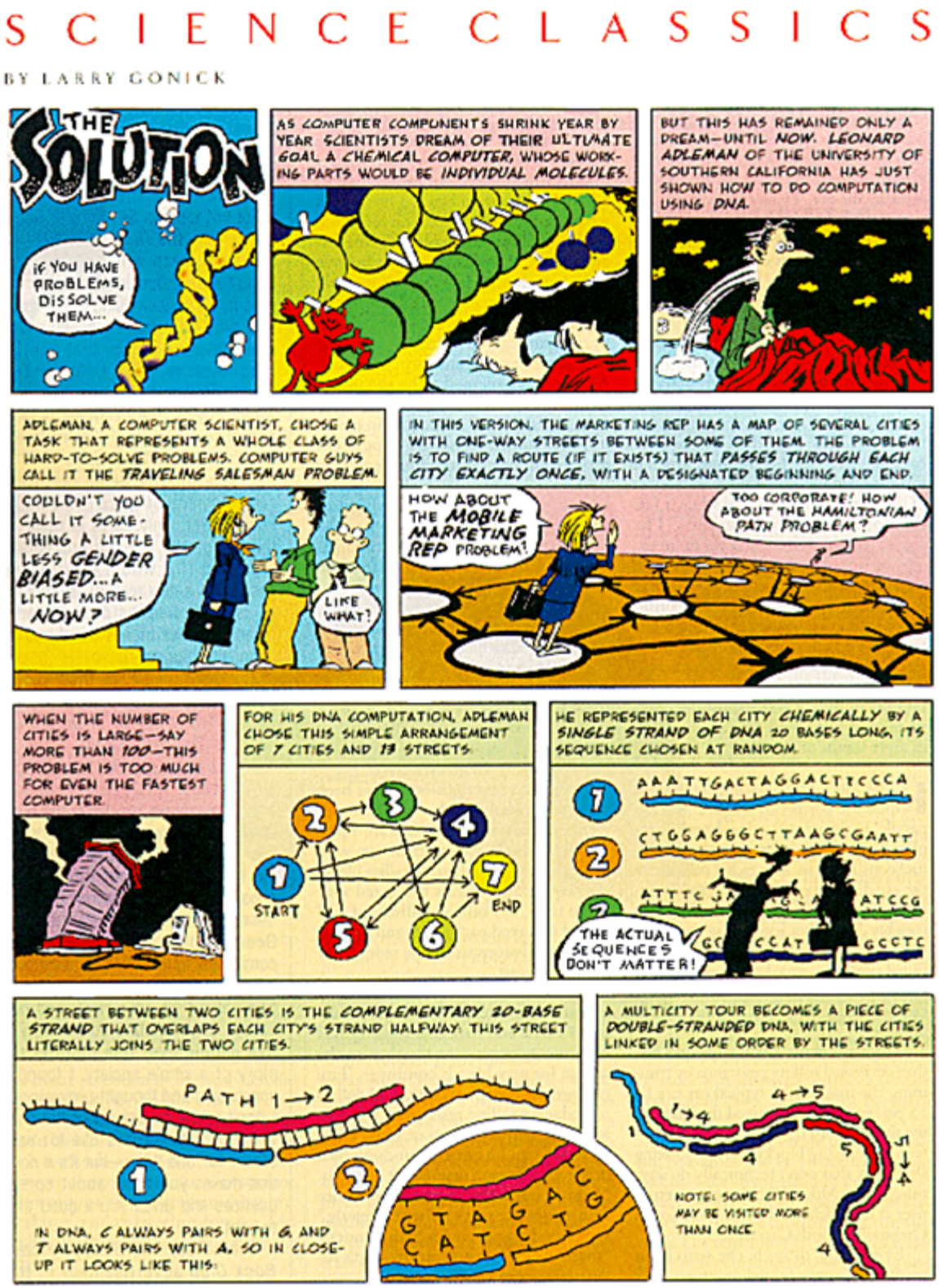
- Adleman solved a seven city problem. What about more cities.
- The complexity of the traveling salesman problem still increases exponentially.
- For Adleman's method the amount of DNA scales exponentially: to solve a 200 city HP problem would take an amount of DNA that weighed more than the earth
- Each step contains statistical errors which grow if the strands become longer, more operations are being performed and less DNA is used per potential solution

Conclusion on DNA computing

- Rather unsophisticated w.r.t. the computational problem, possibly to become more efficient
- Can be performed automatically
- Most genetic operations are relatively cheap today, they become faster and smaller
- DNA chips are in operation
- Microelectromechanical systems (MEMS)

L.M. Adleman: Molecular computation of solutions to combinatorial problems. Science 226 (1994), 1021-1024.

G. Paun: Computing with Bio-Molecules, Springer-Verlag, 1998.



Discover magazine published an article in comic strip format about Leonard Adleman's discovery of DNA computation. Not only entertaining, but also the most understandable explanation of molecular computation I have ever seen.

Deepthi Bollu

Emergence

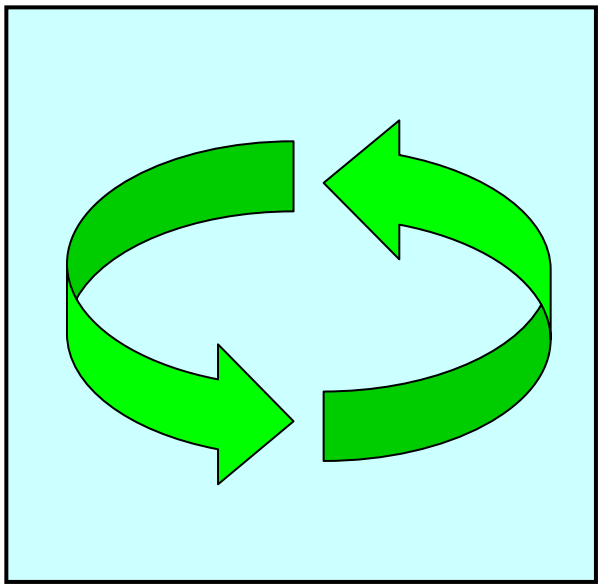
- Radical novelty (features not previously observed in systems are brought about by interaction of the parts)
- Coherence or correlation (meaning integrated wholes that maintain themselves over some period of time)
- A global or macro "level" (i.e. there is some property of "wholeness")
- it is the product of a dynamical process (it evolves)
- it is "ostensive" (it can be perceived).
- For good measure, Goldstein throws in supervenience (Tony Bell: submergence) -- downward causation.

(Corning 2002)

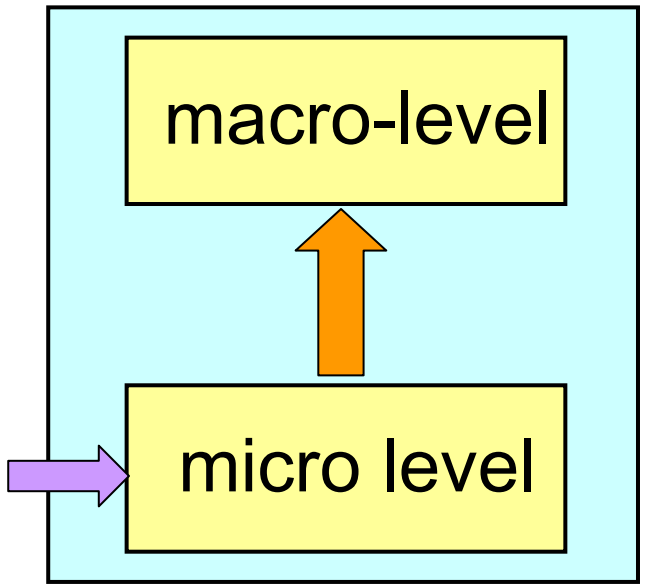
Self-Organization

- Prototypical concept in physics: Phase transitions
- Structure formation
- Self-organization can be a mechanism of emergence, but there is also
  - emergence without self-organization (e.g. temperature as a property of many unordered particles) and
  - self-organization without emergence (e.g. simple flocking behaviour)

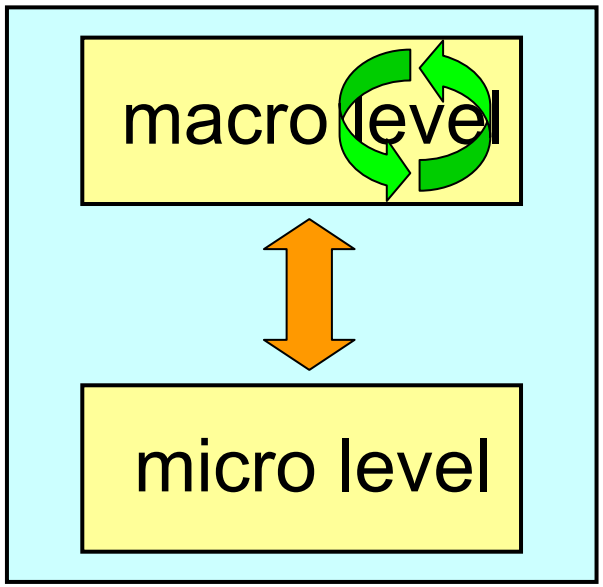
Self-Organisation and Emergence



SO without emergence



emergence without SO



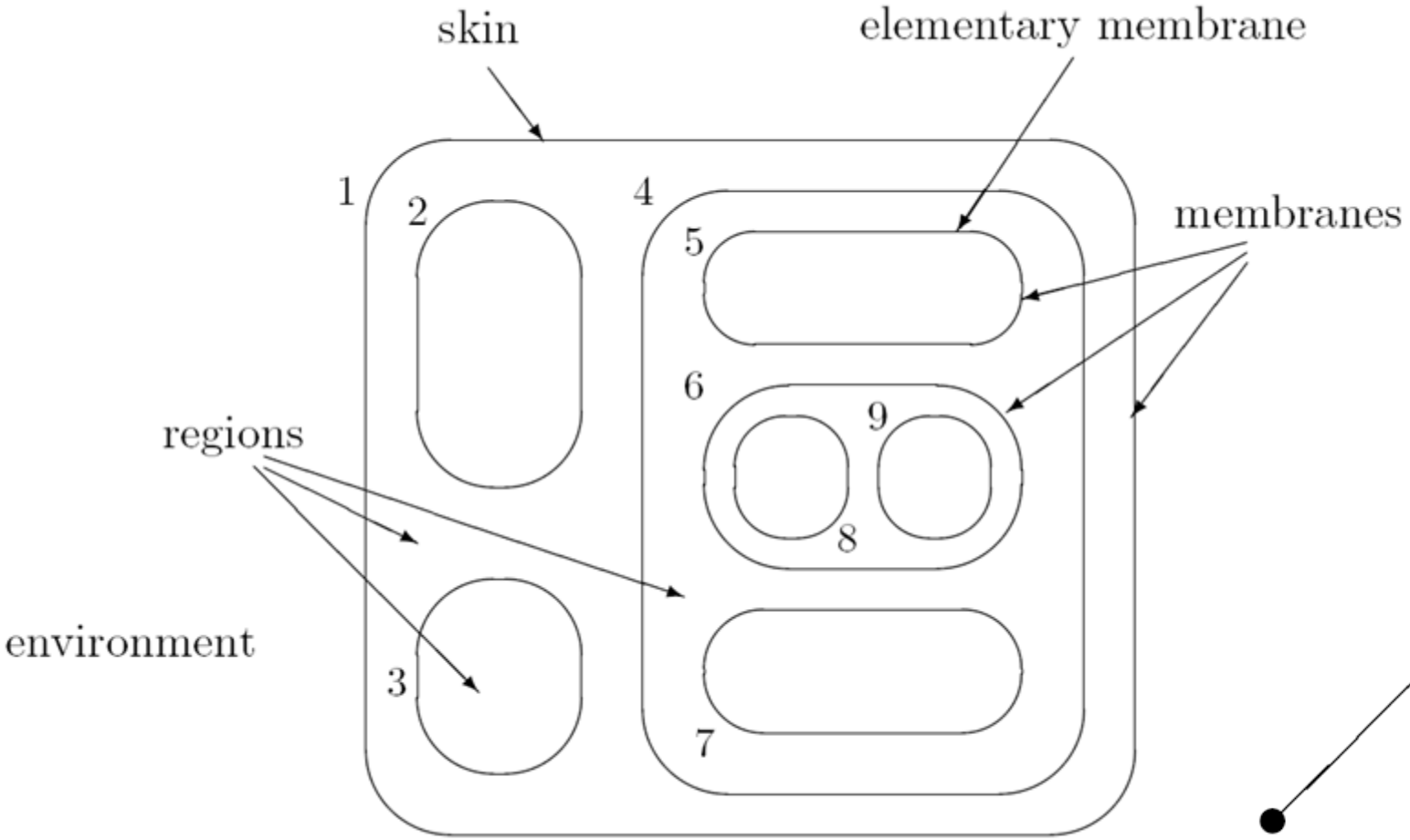
both SO and emergence

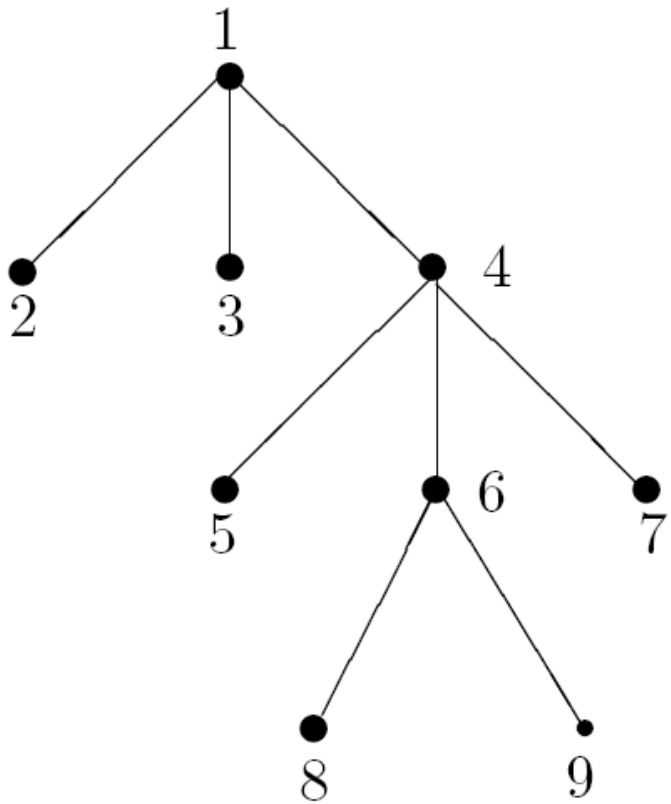
SO, Emergence and Computing

- Elements follow simple physical, chemical, or biological rules.
- Computation is a an aspect that is apparent only at a scale larger than that of the elements
  (a classical computer is not usefully described by quasiparticles in a semiconductor)
- The interplay of the elementary and the computational level introduces a tradeoff between robustness and efficiency
- Computation is brought about by cooperation, selection and amplification of low-level effects

Membrane Computing

- **Membrane computing** is an area that seeks to discover new computational models from the study of the cellular membranes.
- It not so much the task of creating a cellular model but to derive a computational mechanism from processes that are know to proceed in a cell.
- deals with distributed and parallel computing models, processing multisets of symbol objects
- The various types of membrane systems have been formalized as P systems.





$$[_1\ [_2\ ]_2\ [_3\ ]_3\ [_4\ [_5\ ]_5\ [_6\ [_8\ ]_8\ [_9\ ]_9\ ]_6\ [_7\ ]_7\ ]_4\ ]_1$$

$$=$$

$$[_1\ [_3\ ]_3\ [_4\ [_6\ [_8\ ]_8\ [_9\ ]_9\ ]_6\ [_7\ ]_7\ [_5\ ]_5\ ]_4\ [_2\ ]_2\ ]_1$$

P systems

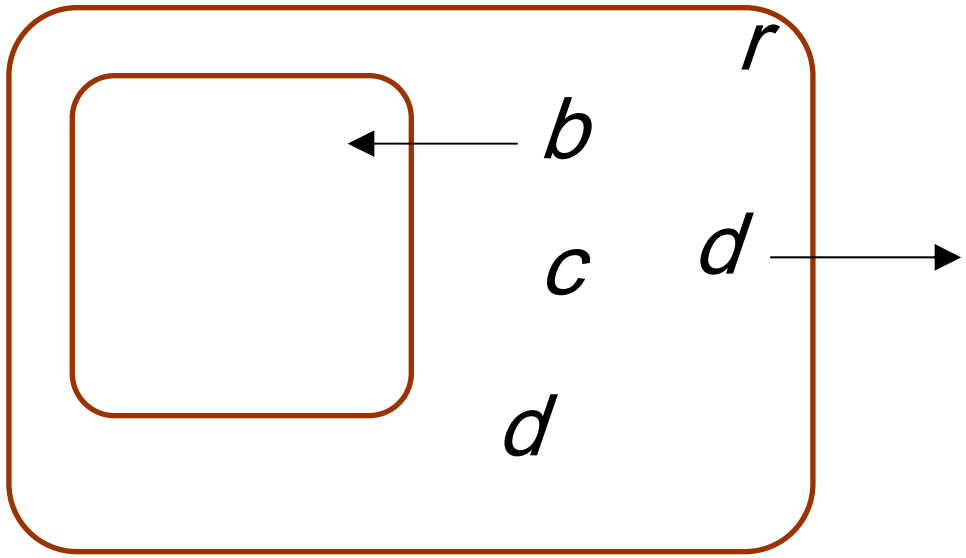
- after Gheorghe Paun: *Computing with Membranes*, (1998)
- A computational rather than a biological model
- Membrane Computing looks at the whole cell structure and functioning as a computing device
- Membranes play a fundamental role in the cell as filters and separators
- Modeling the living cell is beyond the purpose of Membrane Computing

P systems

- A membrane structure formed by several membranes embedded in a unique main membrane
- Multi-sets of objects placed inside the regions delimited by the membranes (one per each region)
- The objects are represented as symbols of a given alphabet (each symbol denotes a different object)
- Sets of evolution rules associated with the regions (one per each region), which allow the system
  - to produce new objects starting from existing ones
  - to move objects from one region to another

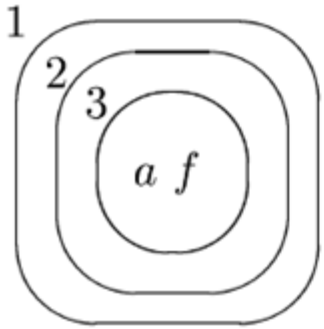
P systems

- Evolution rule of region *r* : $ca \rightarrow cb_{in}d_{out}d_{here}$
  "a copy of object *a* in the presence of a copy of the catalyst *c* is replaced by a copy of the object *b* and 2 copies of the object *d*" and
  "*b* enters the inner membrane of region *r*",
  "one copy of *d* is leaves region *r*"
  and "one copy of *d* remains in *r*."

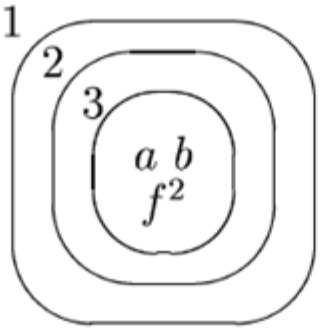


Computation in a P-system

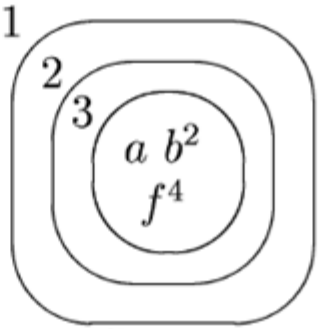
- Initial configuration: Construct a membrane structure and place an initial multi-set of objects inside the regions of the system.
- Apply the rules in a non-deterministic parallel manner: in each step, in each region, each object can be evolved according to some rule
- Halting if a configuration is reached where no rules can be applied
- The result is the multi-sets formed by the objects contained in a specific output membrane
- A non-halting computation yields no result
- Example: Assume the environment is reduced to some specific input objects. Now if the system halts in a final configuration, the system has “recognized” this input.



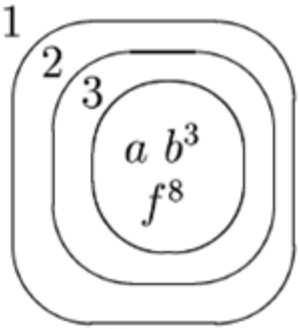
Initial config.



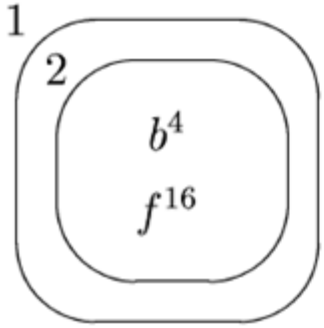
Step 1



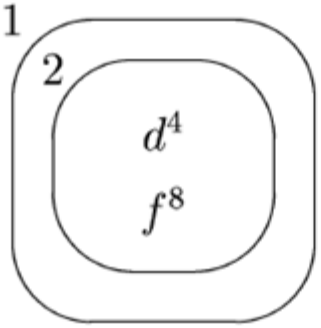
Step 2



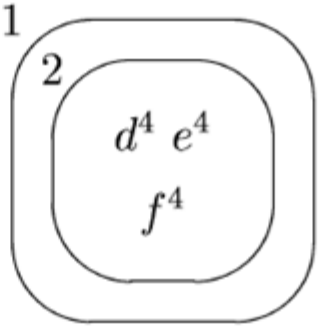
Step 3



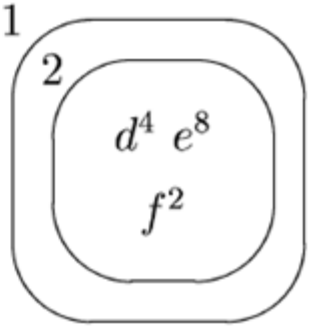
Step 4

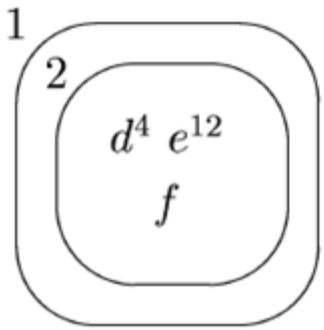


Step 5

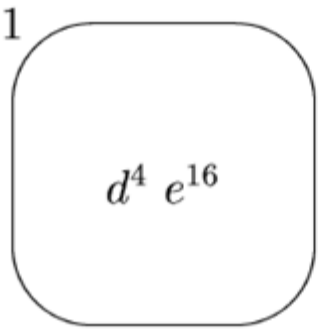


Step 6

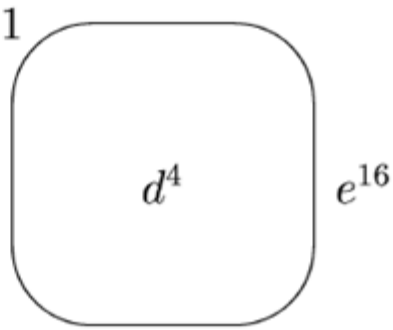




Step 8



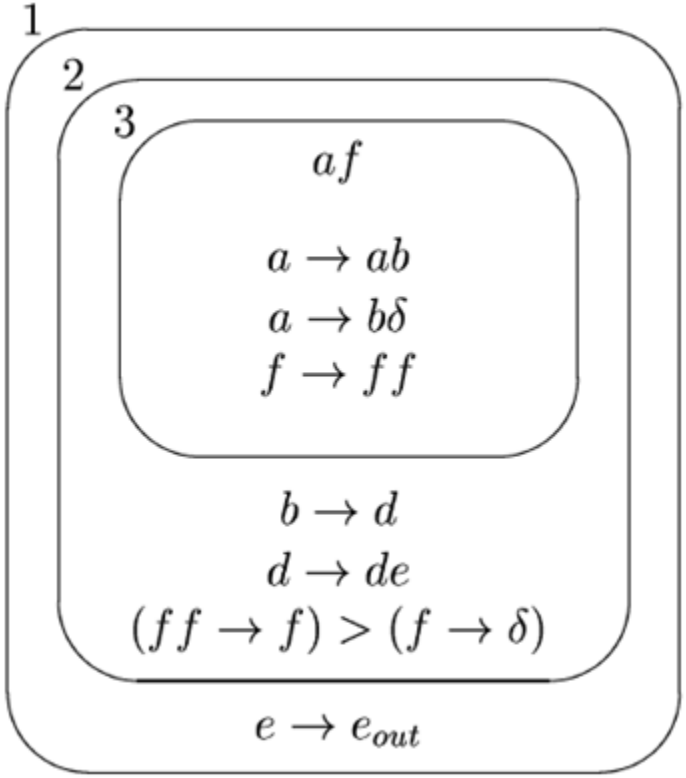
Step 9



Step 10

rules:

δ: dissolutor

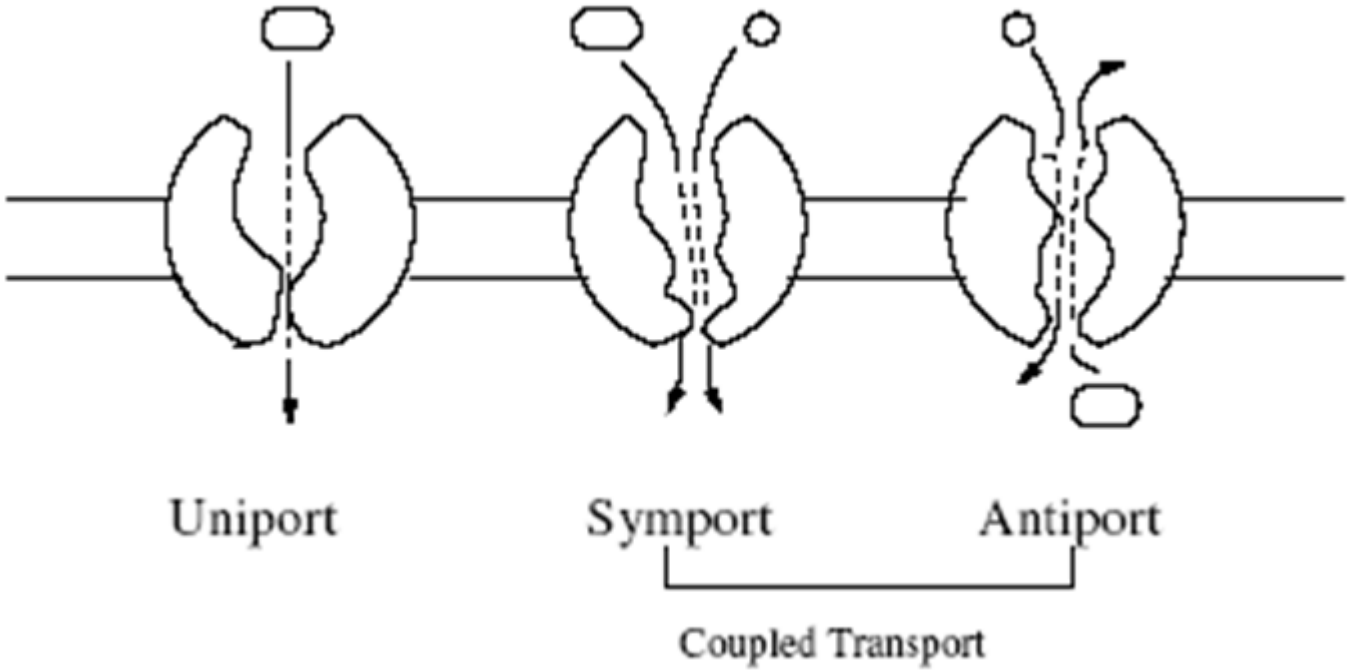


More operators

- membrane dissolution
- priorities, reaction rates
- catalysts
- Bi-stable catalysts
- membrane permeability
- active membranes: creation, deletion, doublication

- P-systems with catalysts are computationally universal

Communicative P-systems

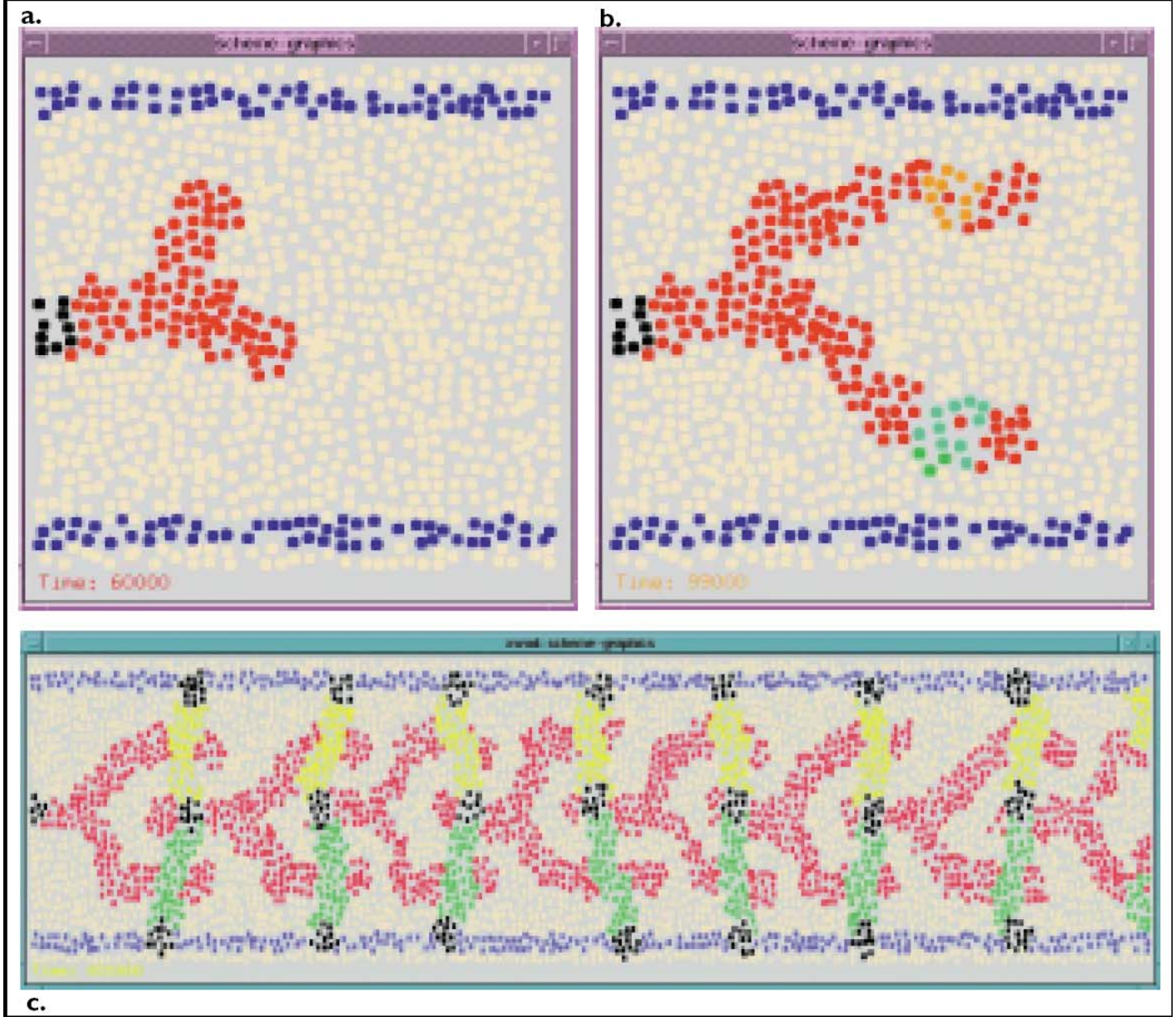


P-systems with symport/antiport are computationally universal (even without chemical reactions)

Amorphous Computing

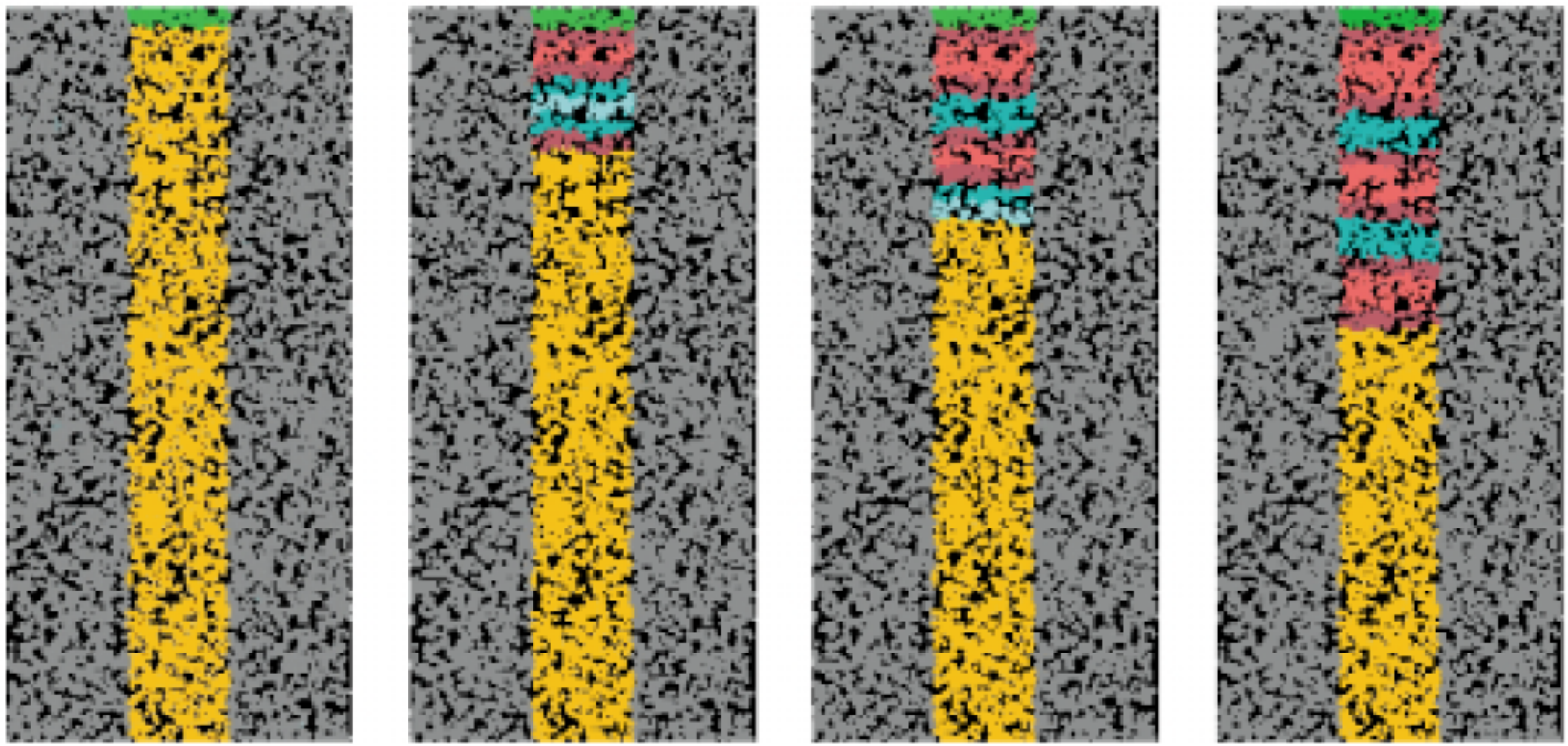
- Implemented by redundant, potentially faulty, massively parallel devices.
- Devices having limited memory and computational abilities.
- Devices being asynchronous.
- Devices having no a priori knowledge of their location.
- Devices communicating only locally.
- Exhibit emergent or self-organizational behavior (patterns or states larger than an individual device).
- Fault-tolerant, especially to the occasional malformed device or state perturbation.

Figure 2. Evolution of a complex design—the connection graph of a chain of CMOS inverters—being generated by a program in Coore's growing-point language. (a) An initial "poly" growth divides to form two branches growing toward "Vdd" and "ground." (b) The branches start moving horizontally and sprout pieces of "diffusion." (c) The completed chain of inverters.



Amorphous Computing

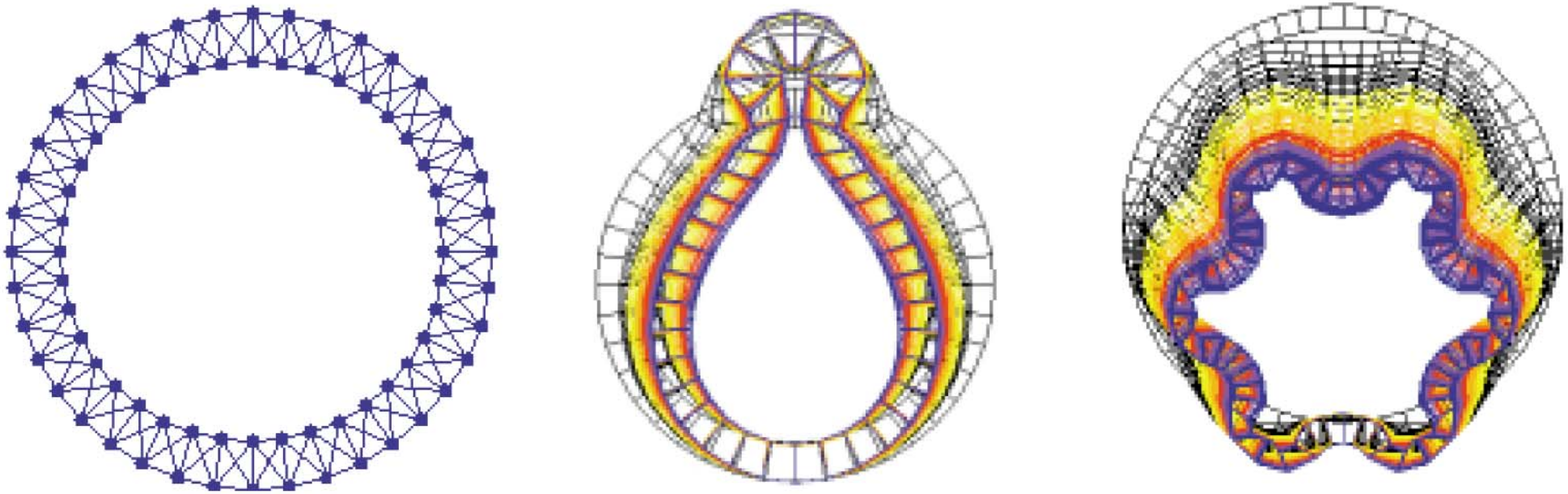
Figure 3. A pattern of alternating bands produced by marker propagation with the aid of Weiss's programming model.



www.eecs.harvard.edu/ssr/papers/cacm00-abelson.pdf

Amorphous Computing

Figure 4. Control of shape changes in a ring of cells, based on the mechanical cell models of [10]. Each cell has a simple programmed behavior and reacts to stresses in its neighbors.



www.eecs.harvard.edu/ssr/papers/cacm00-abelson.pdf

Conclusion

- Nature is the major source of inspiration
  - Natural phenomena can be translated into computing paradigms
  - Natural processes can be (and are) used as carriers of computational operations
- In addition to electronic phenomena other physical effects are prospectively useful for computation
- Specific problems deserve specific solutions, less specified problems require more general approaches