Bio-inspired Models of Computation

An Introduction

Paolo Milazzo

Dipartimento di Informatica, Università di Pisa, Italy

Pisa – April 3rd, 2009

Introduction (1)

Natural Computing is the study of

- models of computation
- inspired by the functioning of biological systems

Natural Computing is not Bioinformatics

- Bioinformatics is the development and application of Computer Science means to Biology (and Medicine)

A relevant part of Natural Computing exploits methods and means of formal language theory

- Biological mechanisms can be often suitably described by rewrite rules

Introduction (2)

The main aims of bio-inspired models of computation are:

- to propose new unconventional computing architectures
- (in the context of formal language theory) to study new classes of formal languages
- to propose new programming paradigms

Typical results on bio-inspired models of computation are:

- Universality (that is Turing completeness)
- Existence of polynomial solutions to NP-complete problems (with exponential workspace)
- Connections with other models of computation (Petri Nets, Lambda calculus, etc...)

Introduction (3)

The aims of this seminar are:

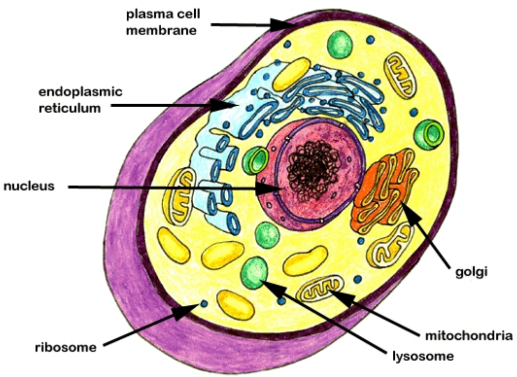
- To give the minimal biological background necessary to understand the mechanisms exploited by bio-inspired models of computation.
- To briefly survey a number of bio-inspired models of computation.
- To give an introduction to P systems (a bio-inspired model of computation)

During this seminar, topics and references for possible student seminars will be given.

Outline of the talk

Outline of the talk

Cells: complex systems of interactive components



- Two classifications of cell:
  - prokaryotic
  - eukaryotic
- Main actors:
  - membranes
  - proteins
  - DNA/RNA
  - ions, macromolecules,...
- Interaction networks:
  - metabolic pathways
  - signaling pathways
  - gene regulatory networks

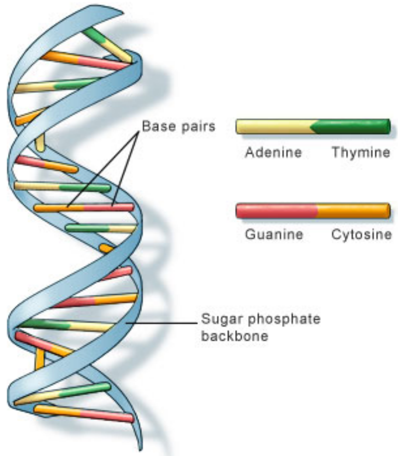
The DNA

The DNA is:

- a molecule
- structured as a string
- over an alphabet of four elements (nucleic acids, bases) denoted A,T,C,G

DNA forms double-stranded helices:

- Base pairing: A–T,C–G
- The *c*omplement of a string is obtained by replacing A with T and C with G, and viceversa
- Two complementary strings form a helic



U.S. National Library of Medicine

Proteins

A gene is a substring of the DNA

- some genes are the "source code" of proteins

A protein is:

- a molecule
- structured as a string
- over an alphabet of twenty elements (amino acids)

Proteins have complex 3D structures related with their functions:

- Catalysis of chemical reactions (enzymes)
- Transport
- Structure
- .....

The central dogma of Molecular Biology

Schematically, in cells we have this flux of information:

$$DNA \xrightarrow{transcription} RNA \xrightarrow{translation} Protein$$

Where the RNA is a molecule structured as a string over the alphabet A,U,C,G (similar to that of DNA)

- It is essentially a copy of the DNA (this motivates the terminology of transcription)

Both transcription and translation can be regulated in order to synthesize proteins only when necessary

On the importance of membranes

Eukariotic cells have a large number of compartments separated by membranes

- this allows a more efficient cell management (based on “divide et impera”)

Membranes have (at least) two fundamental functions:

- as separators of compartments
- as channels of communication between compartments

Outline of the talk

L Systems (1)

Lindenmayer systems (L Systems) are rewriting systems introduced with the aim of modeling the development of multi-cellular organisms.

The main difference with Chomsky grammars is the parallelism:

- in one derivation step all symbols of the string are rewritten

In their simplest form L Systems are defined as

$$G = (V, w, R)$$

where:

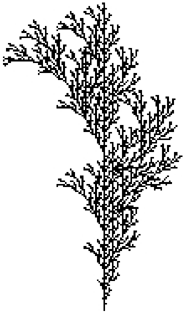
- V is an alphabet
- $w \in V^*$ is the initial state of the system
- R is a finite set of rules $a \longrightarrow v$ with $a \in V$ and $v \in V^*$

L Systems (2)

Examples of L Systems:

- The L System $G = (\{a\}, a, \{a \to aa\})$ generates the language $\{a^{2^n} | n \geq 1\}$ which is not context-free
- The L Systems $G = (\{A, B\}, A, \{A \to AB\ ,\ B \to A\})$ produces

```
n=0:            A
               / \
n=1:          A   B
             /|    \
n=2:        A B     A
           /| |     |\
n=3:      A B A     A B
         /| | |\    |\ \
n=4:    A B A A B   A B A
```



L Systems (3)

L Systems are currently used:

- In architecture: in software tools with 3D visualization of buildings to simulate plants development around the building
- In fractals

Languages genereated by L Systems are recognized by Systolic Automata

Seminar Topic: L Systems

References:

- P. Prusinkiewicz, A. Lindenmayer, **The algorithmic beauty of plants**, Springer-Verlag, 1990.
  one copy available in our library, but also availble for free online
- A. Lindenmayer, **Mathematical Models for Cellular Interaction in Development, Parts I and II**, Journal of Theoretical Biology 18, pp. 280–315, 1968.
- K. Culik, A. Salomaa and J. Gruska, **On a Family of L Languages Resulting from Systolic Tree Automata**, Theoretical Computer Science 23, pp. 231–242, 1983.

Cellular Automata (1)

Cellular Automata consist of a grid of cells, each in one of a finite number of states.

- Time is discrete
- The state of a cell at time t is a function of the states of a finite number of cells (called its neighborhood) at time $t - 1$
- Every cell has the same rule for updating, based on the values in its neighbourhood

The simplest nontrivial Cellular Automaton:

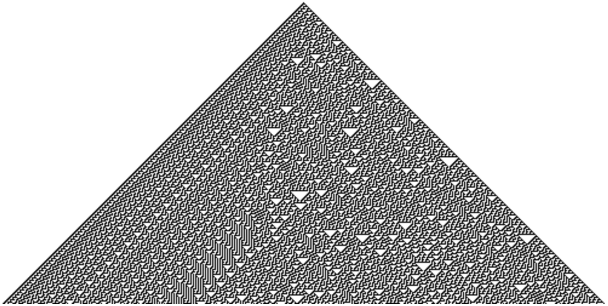
- is one-dimensional with two possible states per cell (an array of bits)
- cell's neighbors are defined to be the adjacent cells on either side of it

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

Cellular Automata (3)

As regards Cellular Automata in 2D:

- NetLogo (http://ccl.northwestern.edu/netlogo/)

Cellular Automata are Turing Complete

- In particular, the game of life has been proved to be universal

Seminar Topic: Cellular Automata

References:

- , **The World Wide Web**, .

Seminar Topic: NetLogo

References:

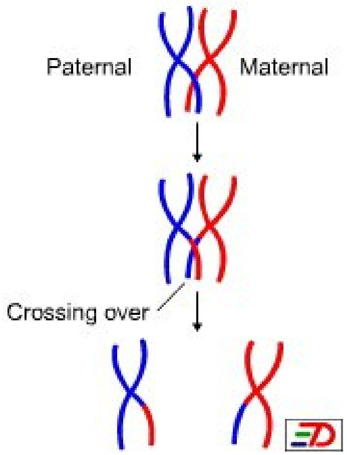
- , **http://ccl.northwestern.edu/netlogo/**, .

Computing by splicing: H Systems (1)

Splicing systems (aka H Systems) are language generating tools based on DNA crossing over operations

- operated in a very specific way by restriction enzymes
- essential for DNA recombination in sexual reproduction

Crossing over operations are used to define a new form of rewrite rule



Computing by splicing: H Systems (2)

The simplest definition of H Systems is

$$H = (V, A, R)$$

where

- V is an alphabet
- $A \subseteq V^*$ is the initial language
- R is a (possibly infinite) set of splicing rules

Splicing rules have the form

$$u\#v\$u'\#v'$$

where u, v, u' and v' belong to V^*

Computing by splicing: H Systems (3)

The language generated by a H System is constructed incrementally by starting from A

- let $u\#v\$u'\#v' \in R$
- let $xuvy$ and $x'u'v'y'$ be two words of the language at step t
- then $xuv'y'$ is a word of the language at step $t+1$

This simple operation on strings is enough to obtain Turing-completeness

Seminar Topic: H Systems

References:

- T. Head, **Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviours**, Bullettin of Mathematical Biology 49(6), pp. 737-759, 1987.
- G. Paun, G. Rozemberg, A. Salomaa, **Computing by splicing**, Theoretical Computer Science 168, pp. 321-336, 1996.
- G. Paun, **DNA computing based on splicing: universality results**, Theoretical Computer Science 231, pp. 275-296, 2000.

DNA Computing (1)

In DNA Computing,

- DNA encodes data
- biotech operations on DNA are used to build a program

Among the operations of membrane computing we have

- Denaturation (separation of a double-stranded helic into two separated sequences)
- Ibridization (pairing, by complementarity, of two sequences)
- Amplification (generation of a number of copies of a DNA fragment)
- Selection by affinity (estraction of DNA molecules that contain a given subsequence)
- ......

DNA Computing (2)

The first DNA Computing experiment has been done by Leonard Adleman

- He solved an instance of the Hamiltonian Path problem with DNA

The algorithm on a graph of N nodes is (roughly) as follows:

1. Each node and each arc of the graph is encoded as a DNA sequence
2. Node i is encoded as the sequence $A_i B_i$ (of length k)
3. The arc connecting nodes i and j is encoded as the sequence $B_i^C A_j^C$, where C denotes string complementation
4. Many copies of the encodings of nodes and arcs are generated and mixed
5. Possible paths are obtained by ibridization of nodes and arcs encodings
6. Finally, if a double-stranded string with a length of $k * N$ and containing all node encodings is present, then such a string represents a solution

DNA Computing and drugs development

DNA Computing will be the topic of a seminar by Prof. Vincenzo Manca

- Hence, we do not propose it as a topic for student seminars

Techniques similar to those of DNA Computing have been proposed for the development of new kinds of drugs:

- the aim is to develop drugs able to sense the presence/absence of some substances and react

Seminar Topic: Drugs development with biomolecular computing

References:

- K. Benenson, T. Paz-Elitzur, R. Adar et al. , **Programmable and autonomous computing machine made of biomolecules**, Nature 414, 2001.
- G. Seelig, D. Soloveichik, D. Yu Zhang, E. Winfree, **Enzyme-Free Nucleic Acid Logic Circuits**, Science 314, 2006.
- K. Rinaudo, L. Bleris, R. Maddamsetti, et al., **A universal RNAi-based logic evaluator that operates in mammalian cells**, Nature Biotech. 25, 2007.
- M.N. Win, C. Smolke, **Higher order cellular information processing with syntetic RNA devices**, Science 322, 2008.

Membrane computing (P Systems)

P Systems are (a class of) formalisms studied in Membrane Computing

P Systems are distributed computing devices inspired by the structure and the functioning of a living cells.

P Systems will be described later.....

Bio-inspired learning and optimization techniques

Biological systems have also inspired learning and optimization techniques

- Neural Networks
- Evolutionary computation
- Ant colonies
- ......

Outline of the talk

P Systems

P Systems are (a class of) formalisms studied in Membrane Computing

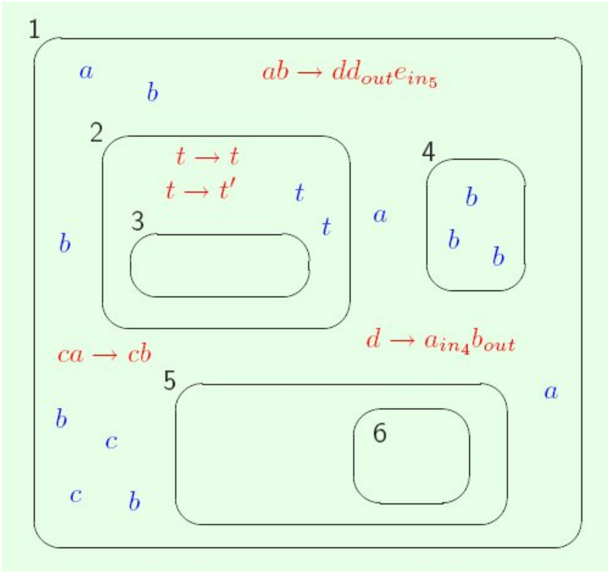
The P Systems web page: `http://ppage.psystems.eu/`

P Systems are distributed computing devices inspired by the structure and the functioning of a living cells.

The key elements of P Systems are:

- Membranes (that create compartments used to distribute computations)
- Multisets (abstractions of chemical solutions that are used as data)
- Evolution (rewriting) rules (abstractions of chemical reactions that are used as programs)

The simplest P Systems



Formal definition of P Systems

A *P System* Π is given by

$$\Pi = (V, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n)$$

where:

- V is an *alphabet* whose elements are called *objects*;
- $\mu \subset \mathbb{N} \times \mathbb{N}$ is a *membrane structure*, such that $(i, j) \in \mu$ denotes that the membrane labeled by j is contained in the membrane labeled by i;
- w_i with $1 \leq i \leq n$ are strings from V^* representing multisets over V associated with the membranes $1, 2, \ldots, n$ of μ;
- R_i with $1 \leq i \leq n$ are finite sets of *evolution rules* associated with the membranes $1, 2, \ldots, n$ of μ.

Evolution rules

An evolution rule $u \rightarrow v$ consists of a multiset of objects u (representing reactants) and a multiset of messages v (representing products). A message may have one of the following forms:

- a_{here}, meaning that object a remains in the same membrane;
- a_{out}, meaning that object a is sent out of the membrane;
- a_{in_l}, meaning that object a is sent into the child membrane l.

The subscript *here* is often omitted.

Evolution rules can be classified into:

- non-cooperative rules: the left-hand side consists of a single object (e.g. $a \rightarrow b^2 d_{out}$)
- cooperative rules: the left-hand side can be any multiset of objects (e.g. $a^2 b \rightarrow b^2 d_{out}$)
  - a particular case of cooperative rules are catalytic rules, namely rules of the form $ca \rightarrow cb^2$ where c belongs to a special set of objects called catalysts.

Maximal parallelism

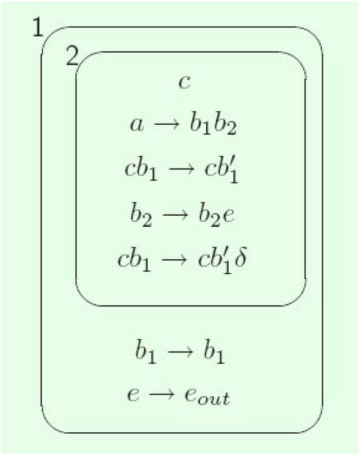
Evolution rules are applied with *m*aximal parallelism:

- More than one rule can be applied (on different objects) in the same step
- Each rule can be applied more than once in the same step (on different objects)
- Maximality means that:

  A multiset of instances of evolution rules is chosen non–deterministically such that no other rule can be applied to the system obtained by removing all the objects necessary to apply the chosen instances of rules.

Example of P System

A P System computing n^2 (with a dissolving rule)



INPUT: a^n inside membrane 2 OUTPUT: e^{n^2} sent out of membrane 1

Universality of P Systems

P Systems with cooperative rules are universal,

- actually, catalytic rules are enough

P Systems with non-cooperative rules only are not universal

In general, universality of a computing model can be proved by showing that another computing model/formalism/system known to be universal can be simulated

- in the case of P Systems, counter machines and matrix grammars with appearence checking are usually exploited

Some variants of P Systems

Variants obtained by considering different types of evolution rules:

- extended with priorities;
- with promoters and inhibitors;
- with dissolution of membranes;
- symport/antiport rules;
- with active membranes;
- ....

Variants obtained by considering graphs rather than trees as membrane structures:

- tissue-like systems;
- spiking neural systems.

P Systems with active membranes

Active membrane means that evolution rules can change the membrane structure of a P System

- In particular, there can be membrane division rules
- $[u]_i \rightarrow [v]_j[z]_k$

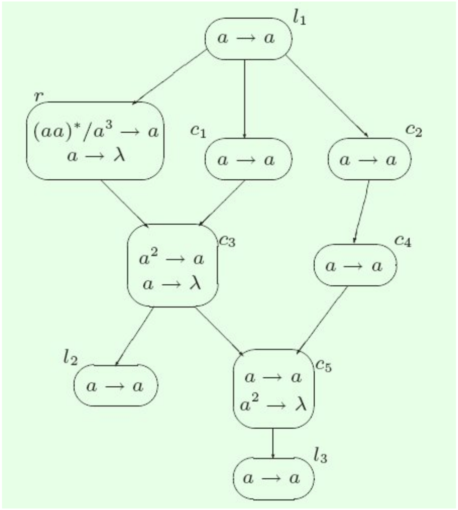
It has been proved that with membrane division it is possible to solve NP-complete problems in polynomial time (but exponential space).

Roughly, the idea is the following:

- Every possible solution is encoded inside a different membrane in a linear number of steps by means of membrane division rules
- Each membrane checks whether the solution it contains is correct (in polynomial time, by def.)

Computational complexity and efficiency aspects of P Systems will be discussed in a seminar by Dr. Claudio Zandron

Spiking Neural P Systems (1)



Spiking Neural P Systems (2)

Although very similar to Petri Nets (that are not universal)

- Spiking Neural P Systems are universal

Seminar Topic: Spiking Neural P Systems

References:

- M. Ionescu, Gh. Paun, T. Yokomori:, **Spiking neural P systems**, Fundamenta Informaticae 71 ,2006.
- Gh. Paun, M.J. Perez-Jimenez, G. Rozenberg, **Spike trains in spiking neural P systems**, Intern. J. Found. Computer Sci. 17, 2006.
- A. Paun, Gh. Paun, **Small universal spiking neural P systems**, BioSystems 90, 2007.

Formal semantics of P Systems

Some formal semantics of P Systems have been defined with different aims;

- to develop interpreters proved to be correct;
- to study causality aspects or P Systems;
- to define behavioural equivalences.

The main difficulty in the definition of a formal semantics is the handling of the maximal parallelism.

Seminar Topic: Formal semantics of P Systems

References:

- O. Andreai, G. Ciobanu, D. Lucanu, **A rewriting logic framework for operational semantics of membrane systems**, Theoretical Computer Science 373, pp. 163–181, 2007.
- N. Busi, **Causality in membrane systems**, Workshop on Membrane Computing (WMC 2007), LNCS 4860, pp. 160–171, 2007.
- R. Barbuti, A. Maggiolo–Schettini, P. Milazzo and S.Tini, **Compositional Semantics and Behavioral Equivalences for P Systems**, Theoretical Computer Science 395, pp. 77–100, 2008.

Conclusions

We have given an overview on (some) bio-inspired models of computation

We have suggested a few topics for student seminars. Other ideas:

- Extensions of P Systems (with time, probabilities, etc...)
- Implementation (in silico) of P systems
- Quantum computing
- Reaction-diffusion computers (chemical computers)
- Self-assembly problems
- Biologically Inspired Networking (self-adapting and self-organizing systems, etc..)
- ....