Nature Motivated Approaches to Computer Science – I.

György Vaszil

University of Debrecen, Faculty of Informatics

Department of Computer Science

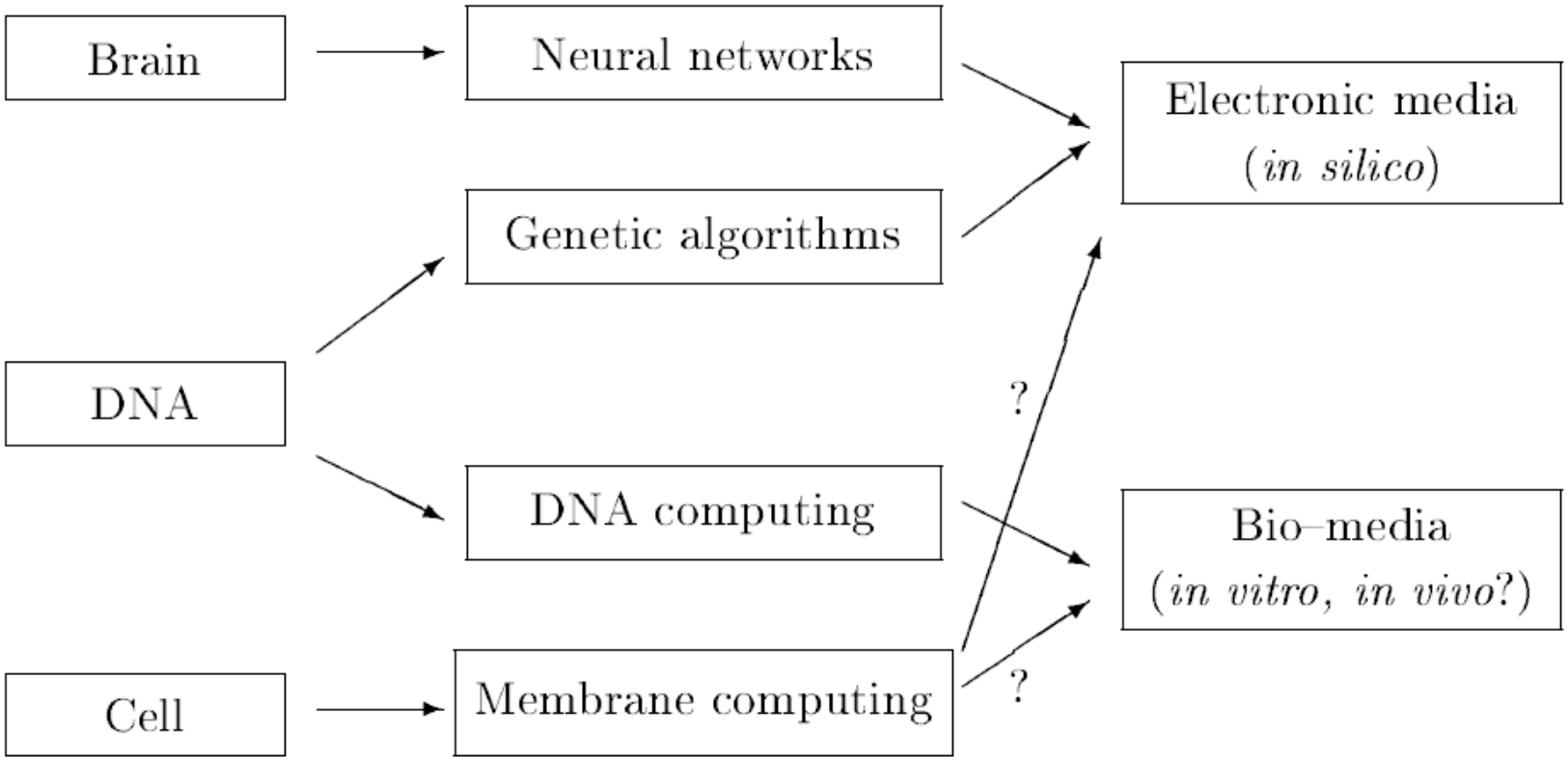
Potsdam, July 2017

What is „Nature Motivated”?

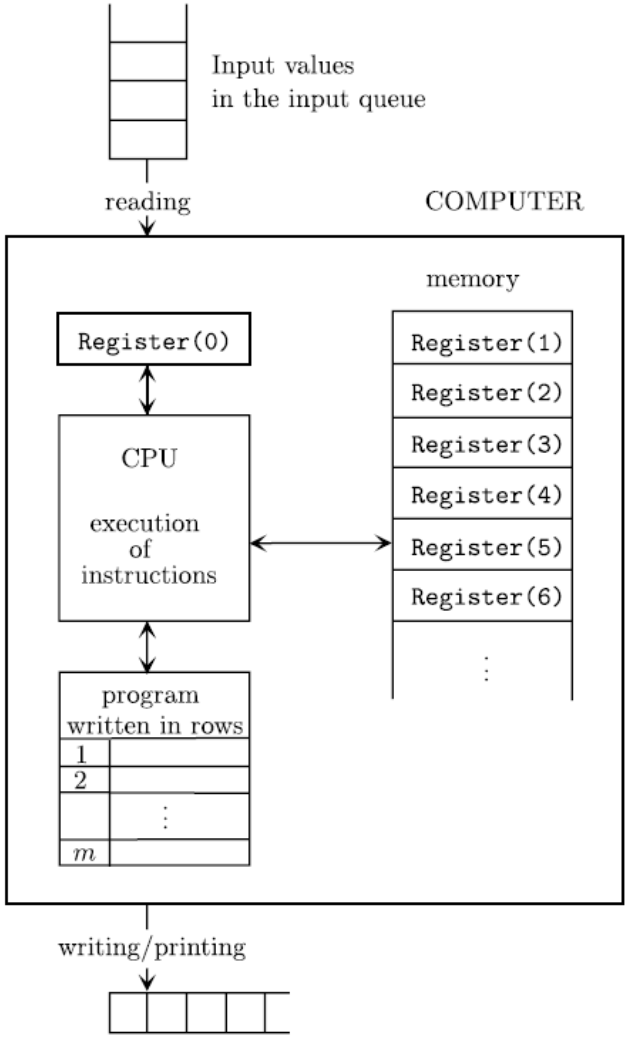
- Theoretical **computational models** constructed with inspiration from nature
- **Computing machinery** with the use of natural processes
- Modelilng and **simulating** natural processes

What is „Nature Motivated”?

Real world/Nature Theoretical model Implementation



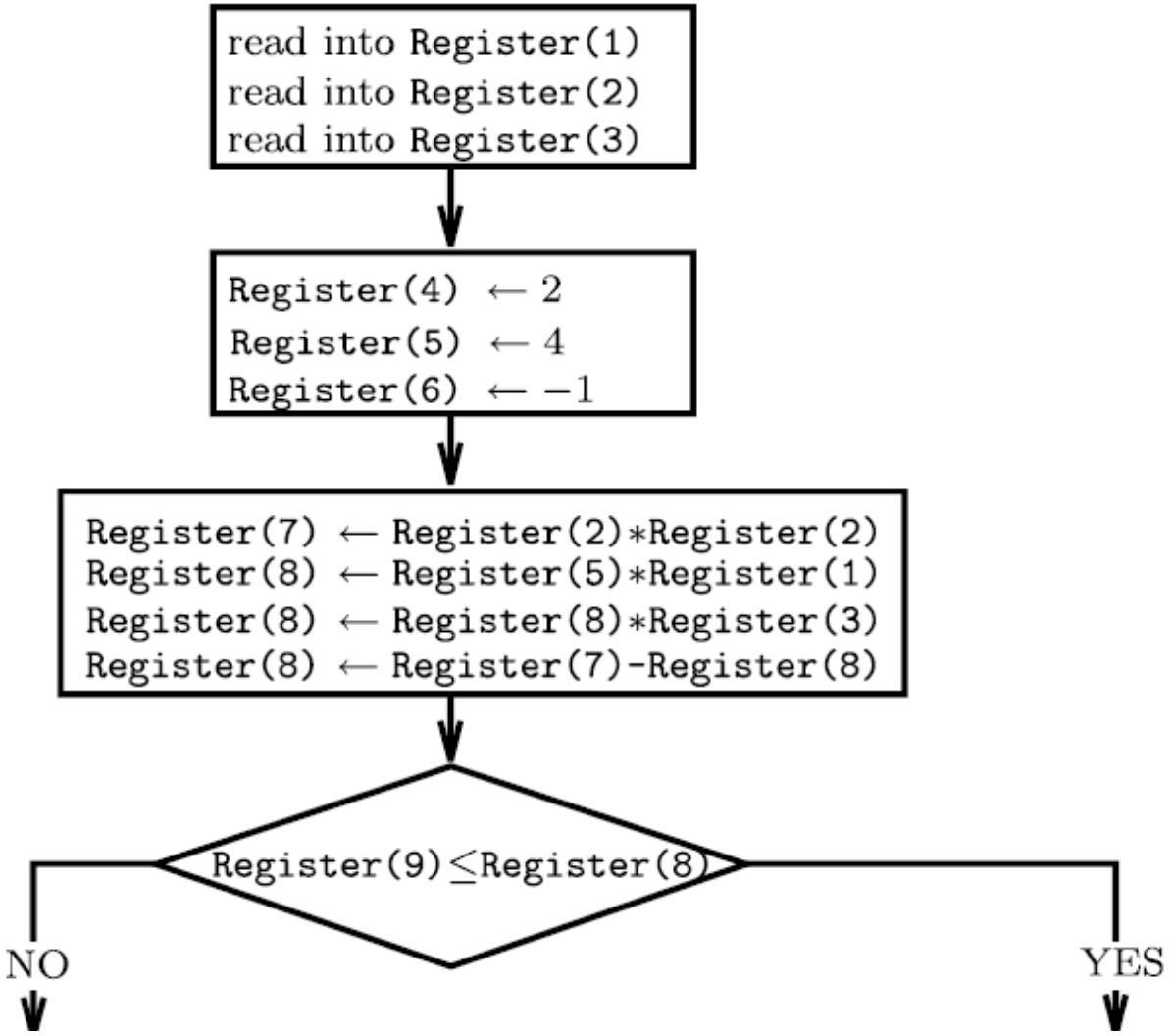
What is a „Computational Model”?



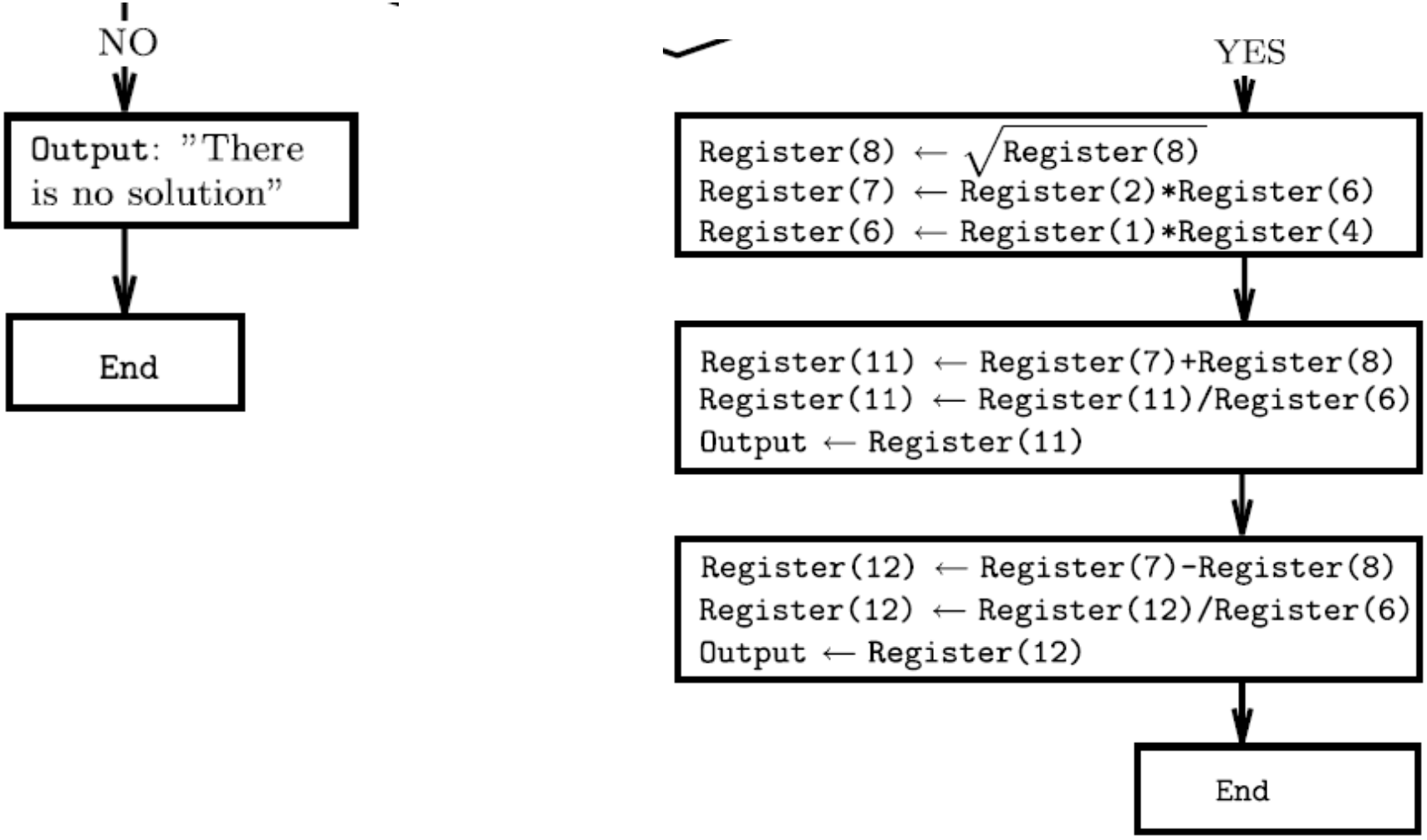
A computational model:
An abstract machine behind the computational process.

- input/output
- method of data storage
- set of instructions
- a method for the application of the instructions
- etc. …

For Example

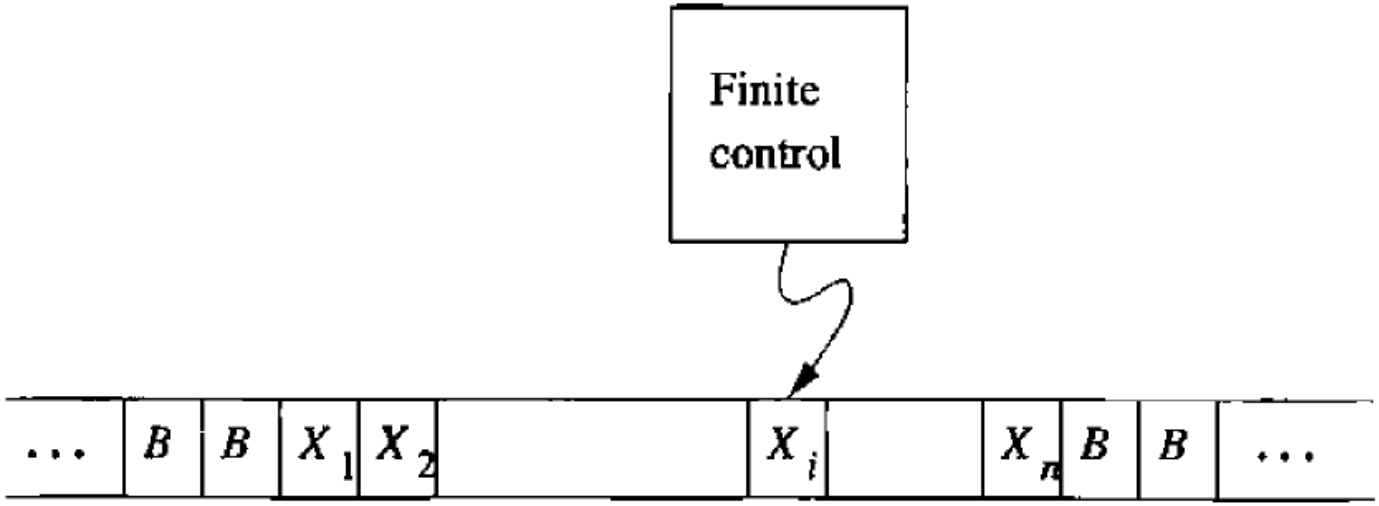

read into Register(1)
read into Register(2)
read into Register(3)
Register(4) ← 2
Register(5) ← 4
Register(6) ← −1
Register(7) ← Register(2)∗Register(2)
Register(8) ← Register(5)∗Register(1)
Register(8) ← Register(8)∗Register(3)
Register(8) ← Register(7)-Register(8)
Register(9)≤Register(8)
NO
YES

Example, continued



- **Random Access Machine**
- **network of logic** gates
- different types of **automata**
- …

Turing machine



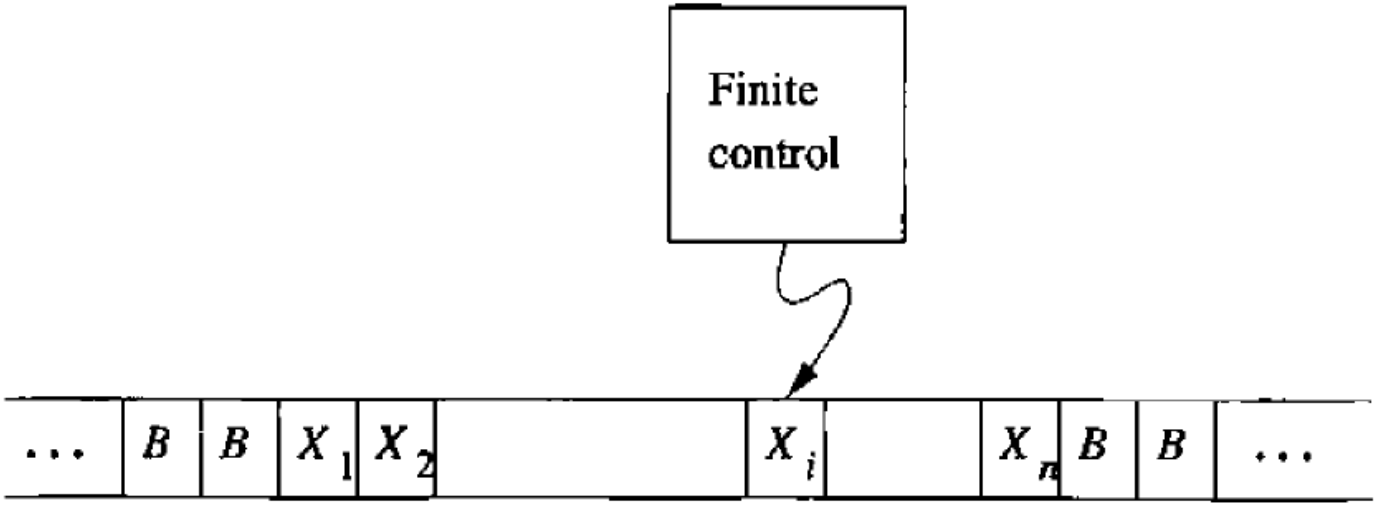
- infinite **tape**, finite tape **alphabet**
- a **read-write head** moving left/right on the tape
- **finite control** (finite set of instructions, finite set of internal states)

Turing machine

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, where

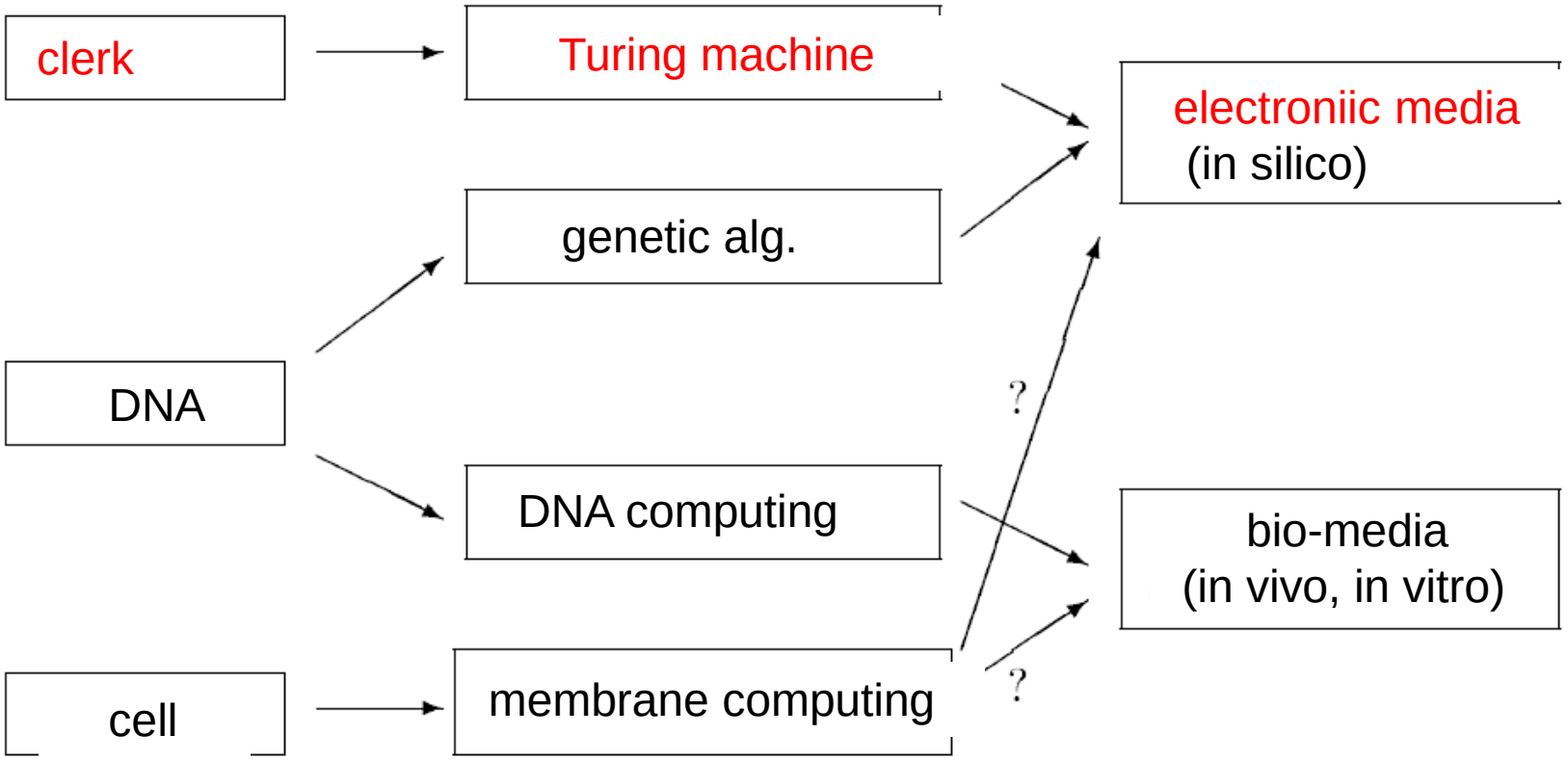
- Q finite set of **states**
- Σ **input** alphabet
- Γ **tape alphabet**
- $\delta : Q \times \Gamma \to 2^{(Q \times \Gamma \times \{L,R\})}$ state **transition** mapping
- $q_0 \in Q$ **initial state**
- $F \subseteq Q$ set of **final states**

Turing machine

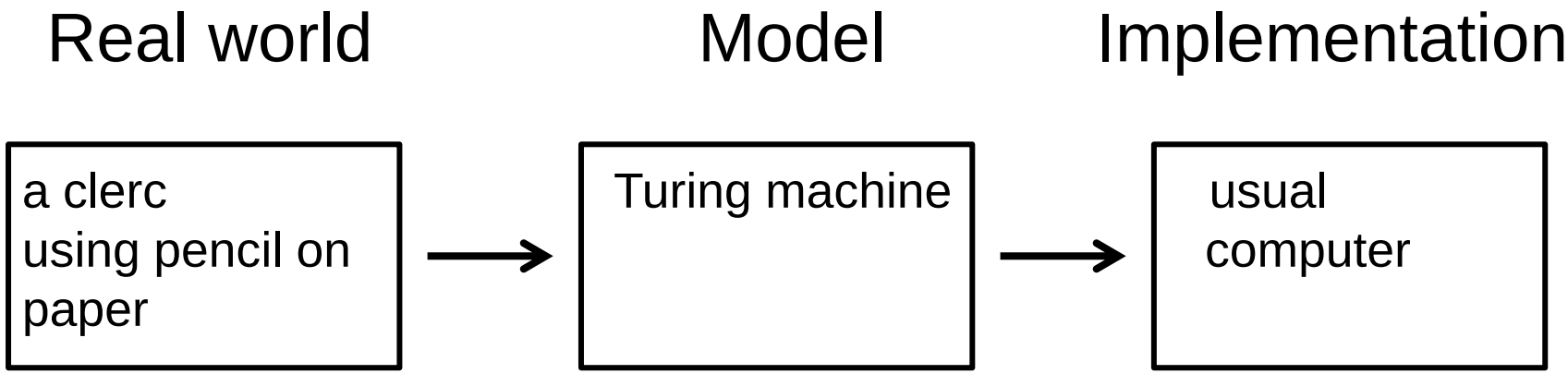


The **Turing machine „can do anything”**. It can compute anything that is algorithmically computable.

Real world | Theoretical models | Implementation



The Turing machine is *not* a nature motivated model



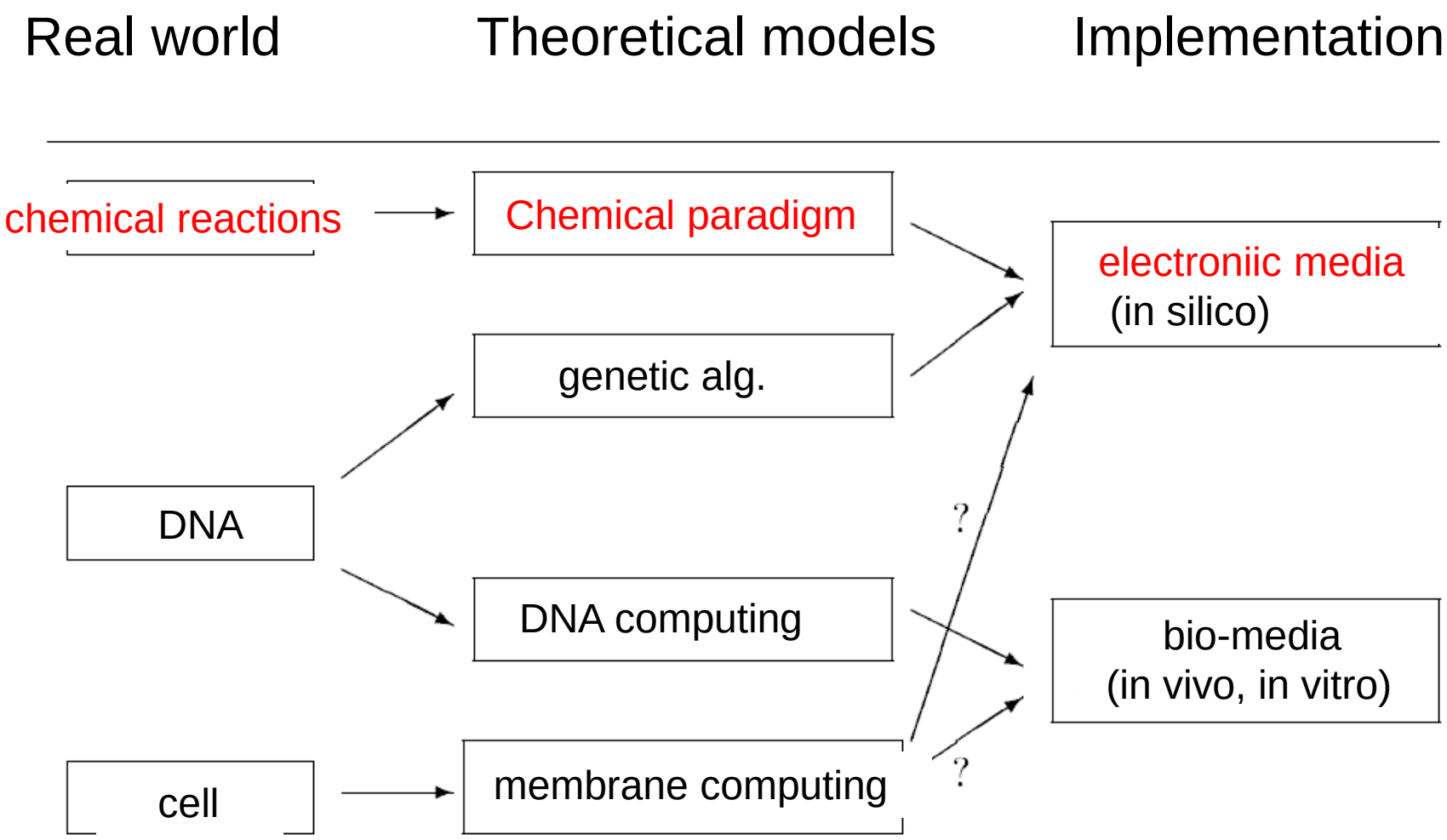
Sequentiality on different levels:

- **data** reading/writing
- the **data structure** itself (tape / string of symbols)
- the order of the execution of **instructions**

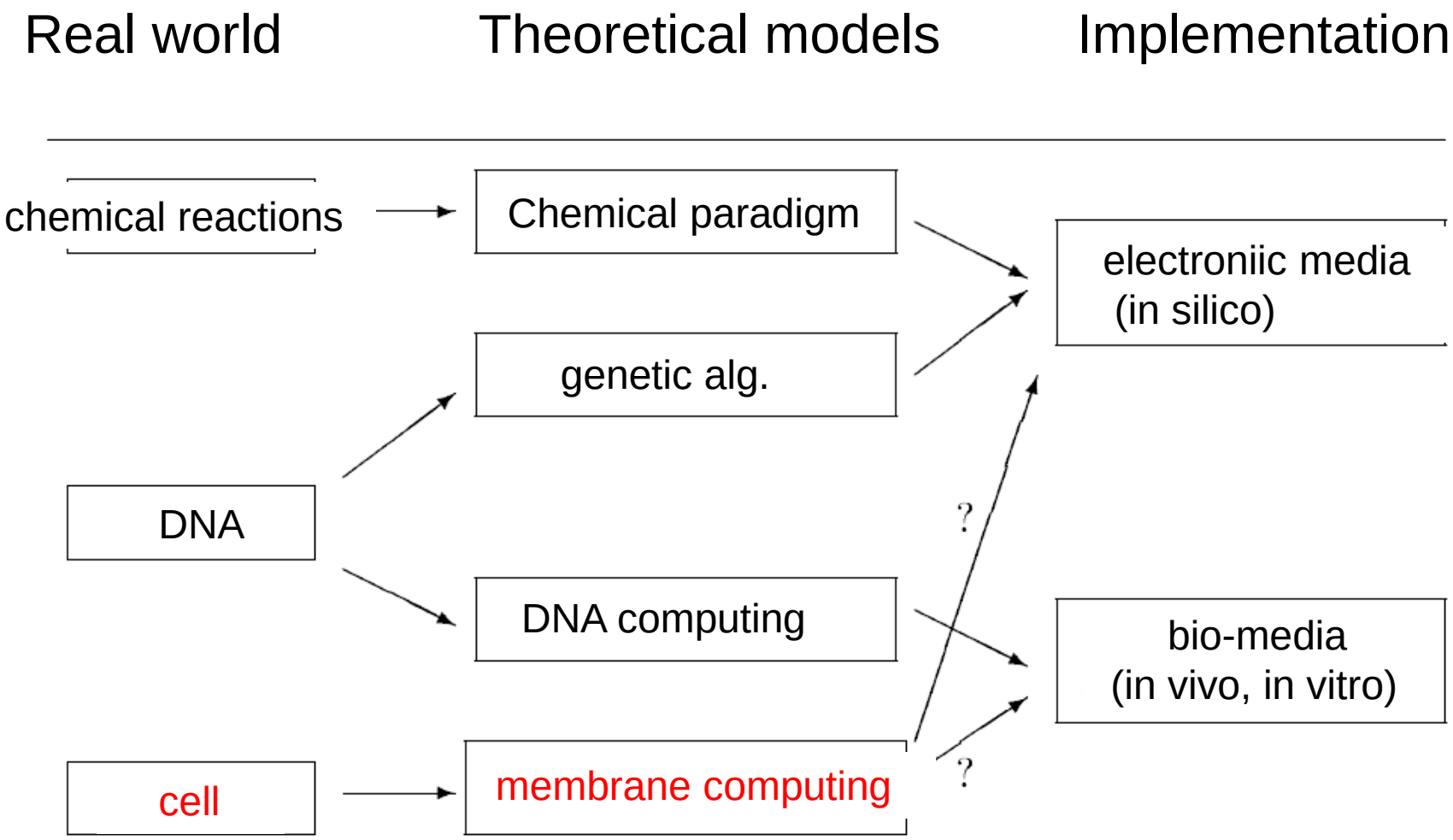
Our topics

- „Chemical” (abstract) computational models
  - What can we learn from chemical processes? (Can we use them for motivating computational models, how?)
  - Chemical programming
  - Membrane systems
- Model and implementation
  - Computing with DNA molecules

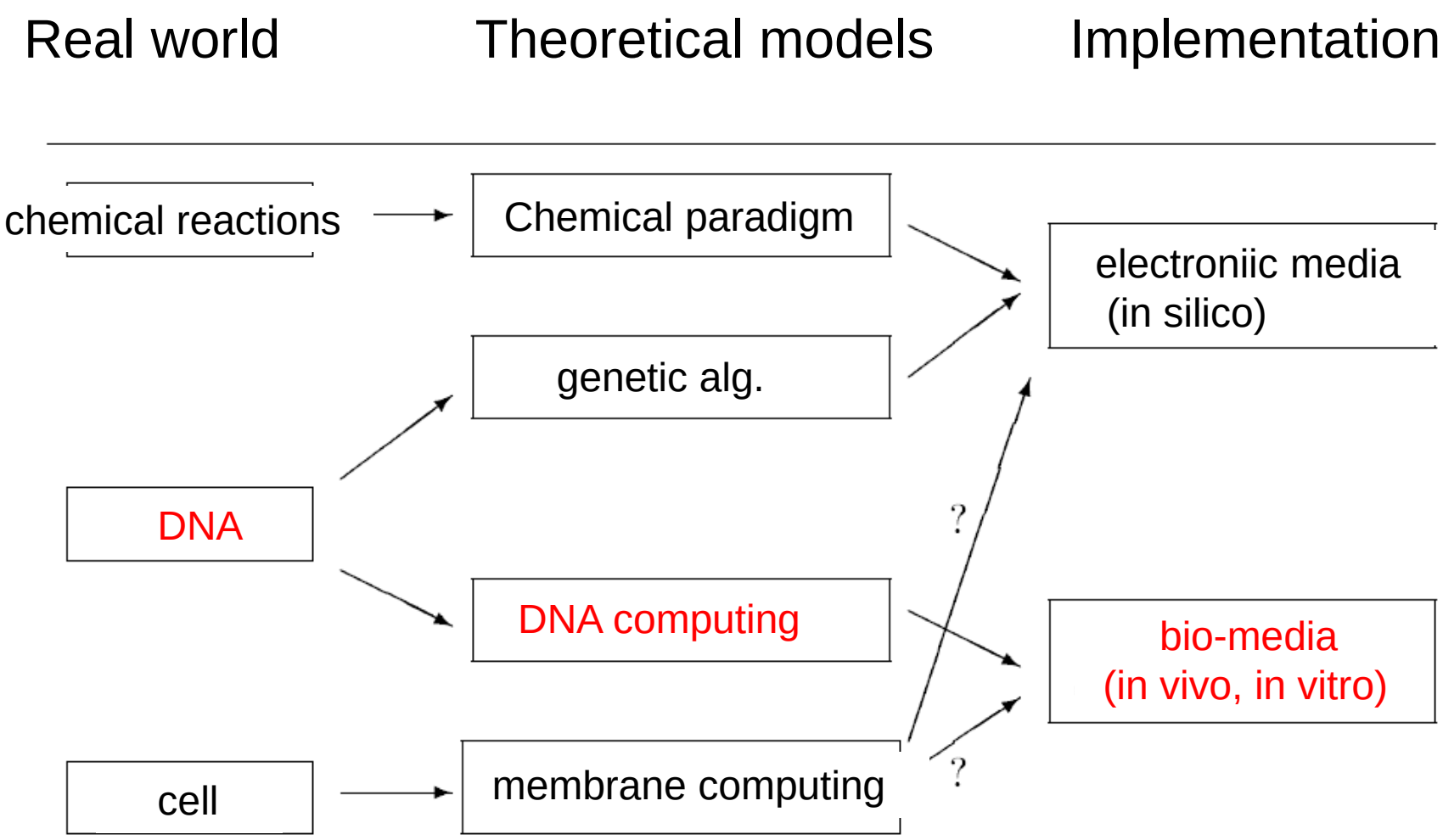
Abstract models - 1



Abstract models - 2



Abstract model + implementation



The chemical computing paradigm

- „Chemistry” as a metaphor
- The goal: free algorithms from those types of sequentiality which is
  - **not** inherently come from the **problem** to be solved
  - the result of the sequentiality of the **computational model** executing the algorithm

J.P. Banatre, D. Le Metayer, 1990

Example – Maximum search

„Conventional” approach:

$$
\begin{aligned}
&maximum \leftarrow set[0] \\
&\textbf{for } loop = 1 \text{ to } n-1 \textbf{ do begin} \\
&\qquad c \leftarrow set[loop] \\
&\qquad \textbf{if } c > maximum \textbf{ then } maximum \leftarrow c \\
&\textbf{end}
\end{aligned}
$$

„Chemical” approach:

while the set contains $\geq$ two elements **do begin**
    select two elements, compare them and remove the smaller
end

The chemical model

- Symbolic **chemical solution** with abstract **molecules**, and rules describing **reactions** between them
- Molecules represent **data**, reactions represent **operations**
- **Brownian motion**, as execution model

More precisely

- **Multisets** of symbols/objects + multiset **rewriting rules**

Abstract machine	Chemistry
Data	Molecule
Multiset	Solution
Parallelism/nondeterminism	Brownian motion
Computation	Reaction

Maximum search - Gamma

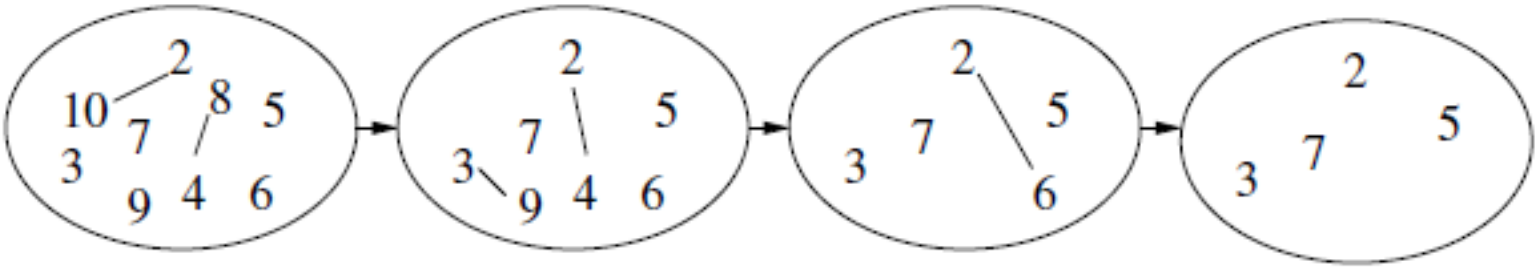
Gamma – General Abstract Model for Multiset Manipulation

$$maxset(s) = \Gamma((R, A))(s) \textbf{ where}$$
$$R(x, y) = x \leq y$$
$$A(x, y) = \{y\}$$

- Multiset as data structure
- Reaction conditions and reaction results
- Parallel application…
- … as long as possible

Example – Primes

$$primes(N) = \Gamma((R, A))(\{2 \ldots N\}) \textbf{ where}$$
$$R(x, y) = multiple(x, y)$$
$$A(x, y) = \{y\}$$



Chemical programming languages an example

„Higher order chemical language” (HOCL):

- **program** = **molecules + reactions**
  + sub-solutions
  contained in an initial **solution**

(higher order: the reactions are also molecules-> they can also evolve)

[Banatre, Fradet, Radenac, 2005]

Chemical programs in HOCL

The **rules** describing the **reactions**:

$$\textbf{replace } P \textbf{ by } M \textbf{ if } C$$

where

- P is a **pattern**
- C is a reaction **condition**
- M is a reaction **product**

For example

The program

$$\{(\textbf{replace } x, y \textbf{ by } x \textbf{ if } x < y),\ 2,\ 7,\ 4,\ 3,\ 6,\ 8\}$$

returns the minimum of 2,7,4,3,6,8.

Another example

Reaction **rules**:

$$\textbf{let}\ \text{multiplier} = \textbf{replace}\ x, \omega\ \textbf{by}\ \omega\ \textbf{if}\ \text{not}(4\ \text{div}\ x\ \text{and}\ 6\ \text{div}\ x)$$
$$\textbf{let}\ \min = \textbf{replace}\ x,\ y\ \textbf{by}\ x\ \textbf{if}\ x < y$$

The **solution** with a sub-solution:

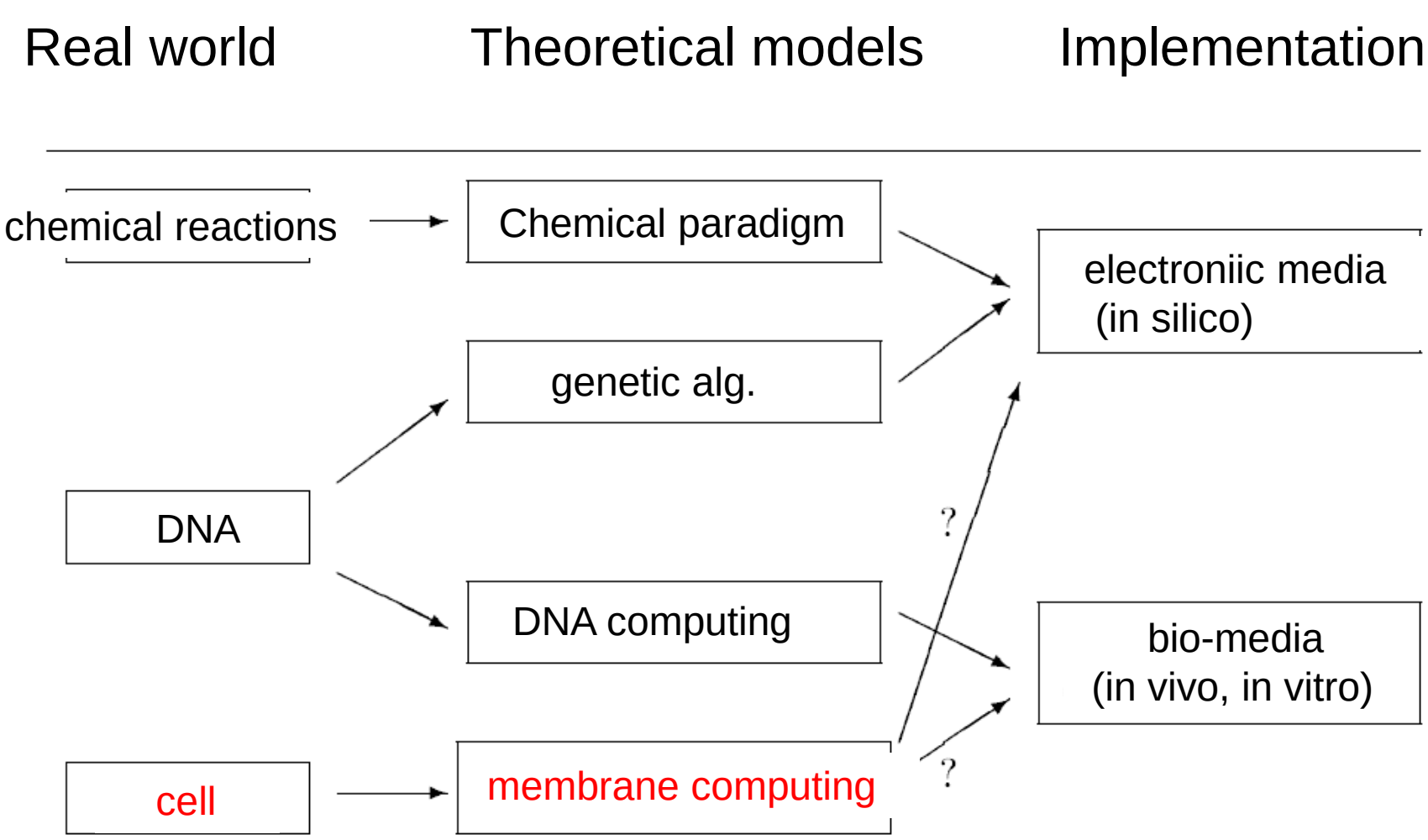
$$\{\min,\ \{\text{multiplier},\ 10,\ 11,\ 12,\ 13,\ 14,\ 15,\ 16,\ 17,\ 18,\ 19,\ 20,\ 21,\ 22,\ 23,\ 24\ \ldots\}\}$$

Result: the least common multiple of 4 and 6.

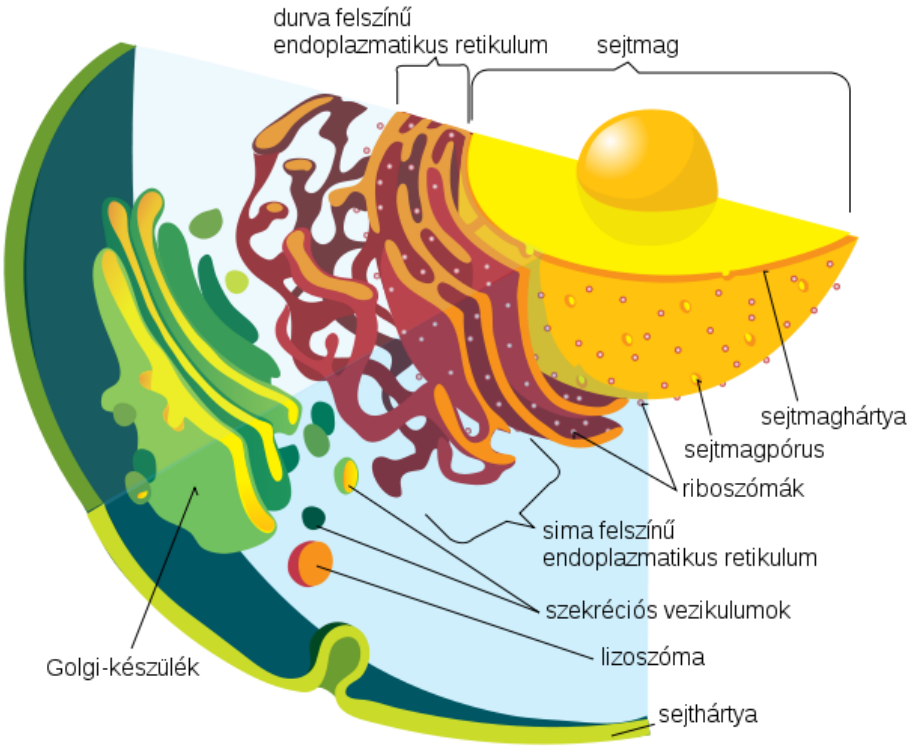
Properties of chemical programs

- M**ultiset** of abstract molecules in a
- **solution**, with
- **reactions** (operations):
  reaction **condition** + reaction **result**.
- **Sub-solutions**: „sub-regions” with their own reaction rules (priority, sequentiality).
- Program execution **ends**, if there are no applicable reactions.

→Natural, free from the forced sequentiality of the physical computer architecture


Real world
Theoretical models
Implementation
chemical reactions
Chemical paradigm
electroniic media
(in silico)
genetic alg.
DNA
?
DNA computing
bio-media
(in vivo, in vitro)
cell
membrane computing
?

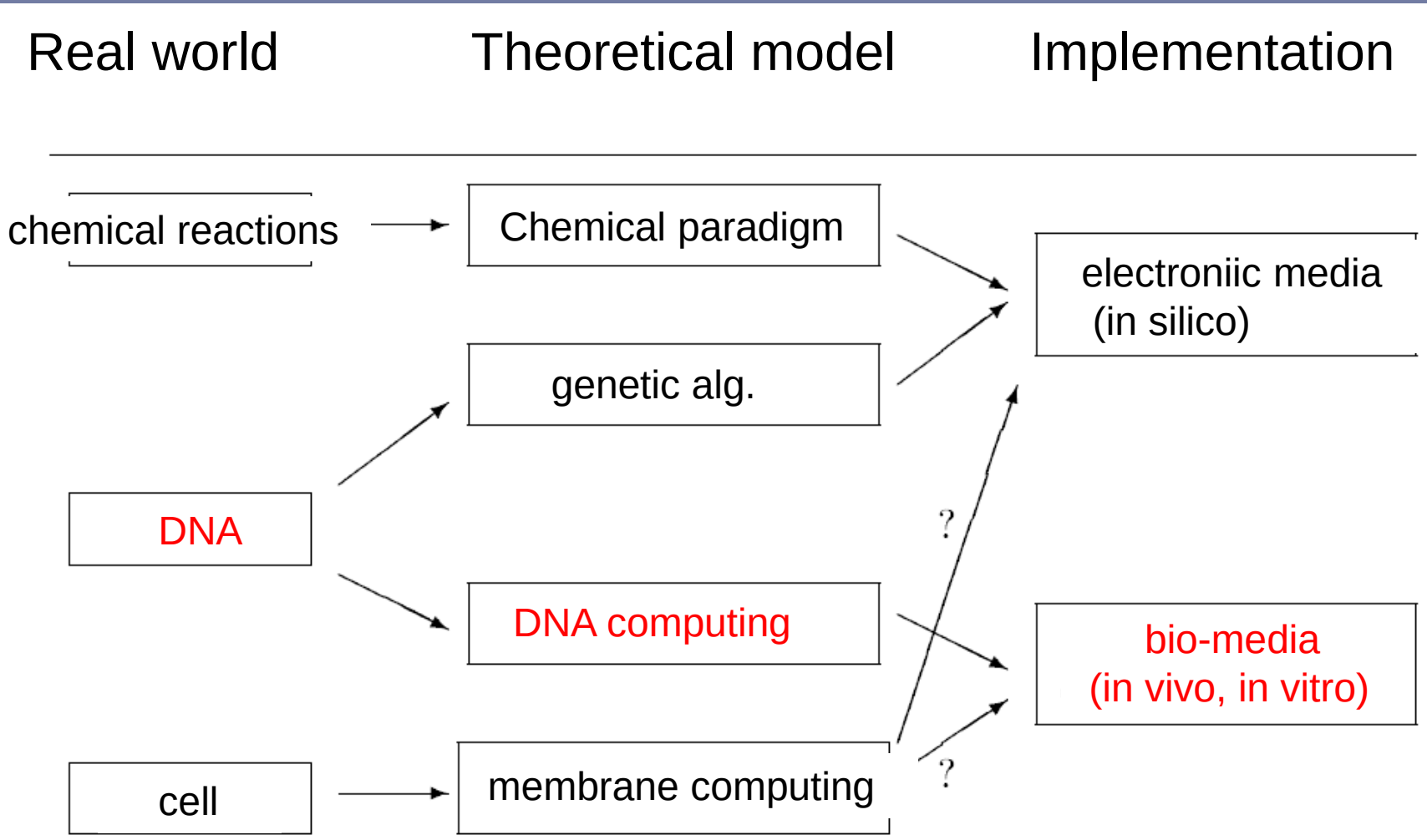
Membrane systems – The biochemical motivation



Cells contain regions

- Regions are enclosed by membranes
- Different regions have different biochemical processes inside
- Membranes also regulate traffic between the regions

More on this next week.


Real world
Theoretical model
Implementation
chemical reactions
Chemical paradigm
electroniic media
(in silico)
genetic alg.
DNA
DNA computing
bio-media
(in vivo, in vitro)
cell
membrane computing
?
?

- Data is not represented by abstract objects, but real DNA molecules, solution in a test tube

→

The possibility of implementation

(Abstract model and implementation in one.)

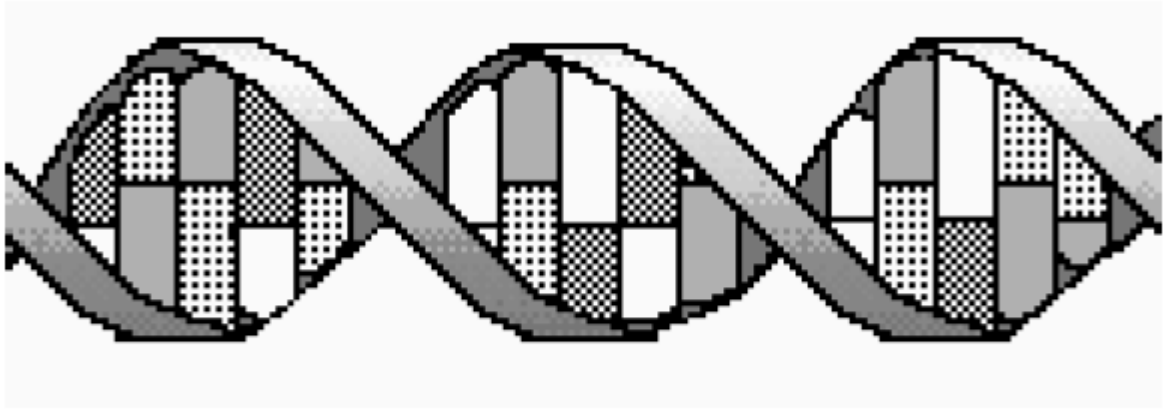
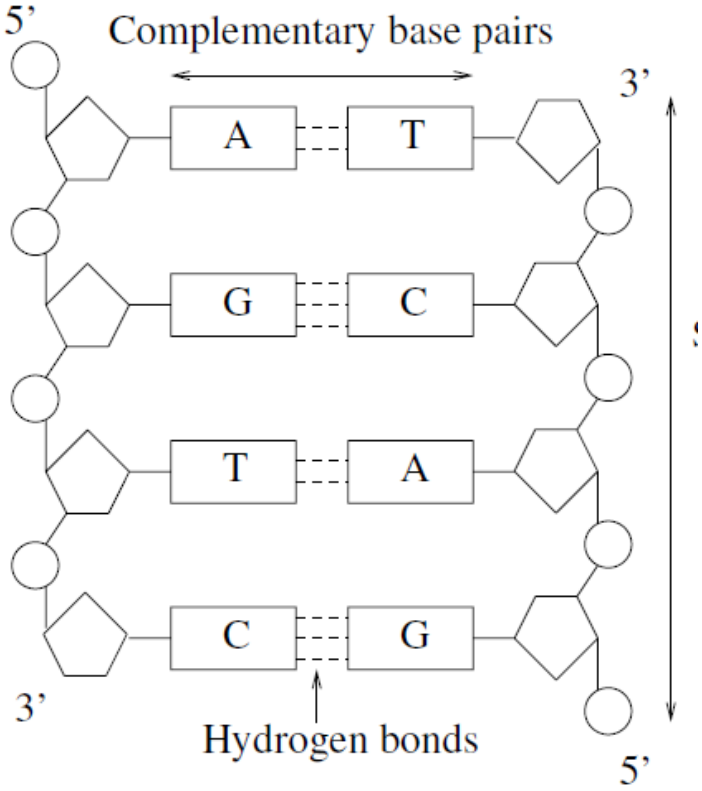


Fig. 1.1. Stylized depiction of DNA double helix



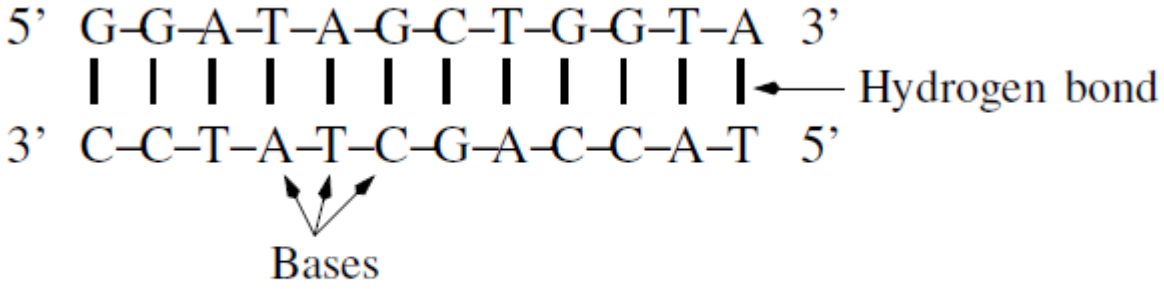


Fig. 1.2. Structure of double-stranded DNA

Operations, laboratory techniques

1. **Synthesis**: Arbitrary base sequences can be produced in millions of copies
2. **Denaturation**: Double strings fall apart to single strings when heating the solution
3. **Hybridization**: Single strings combine to double strings (ligase enzymes)

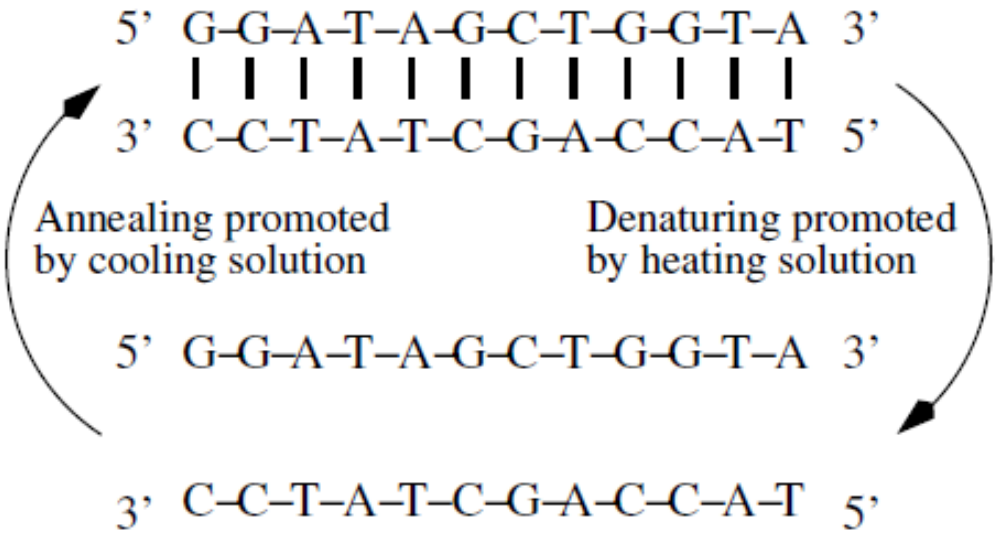


Fig. 1.6. DNA melting and annealing

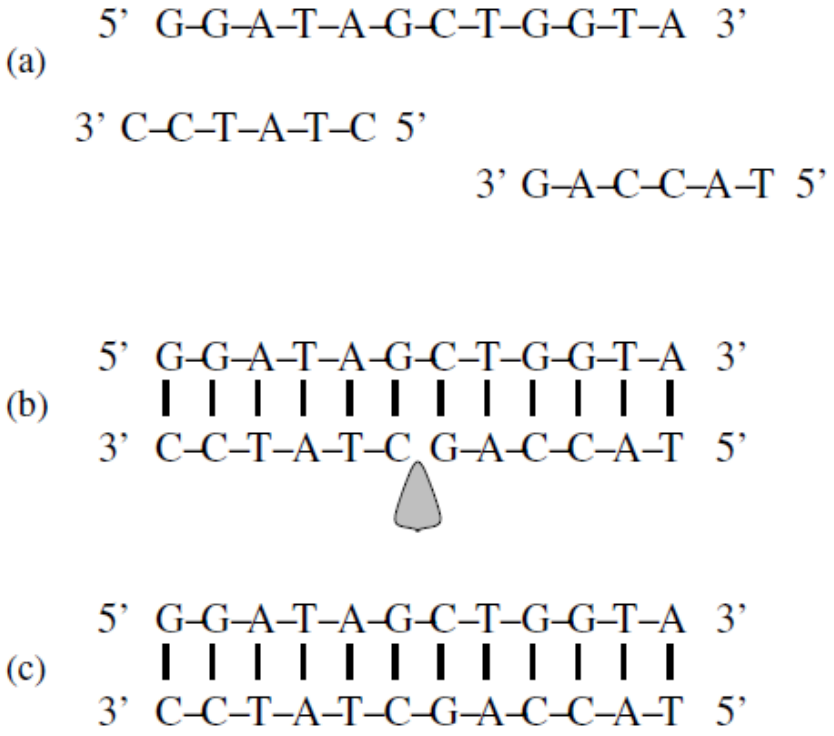


Fig. 1.7. (a) Three distinct strands. (b) Ligase repairs discontinuity. (c) The resulting complex

Operations - 2

4. **Selection**: Removing molecules containing a specific string/sequence of bases

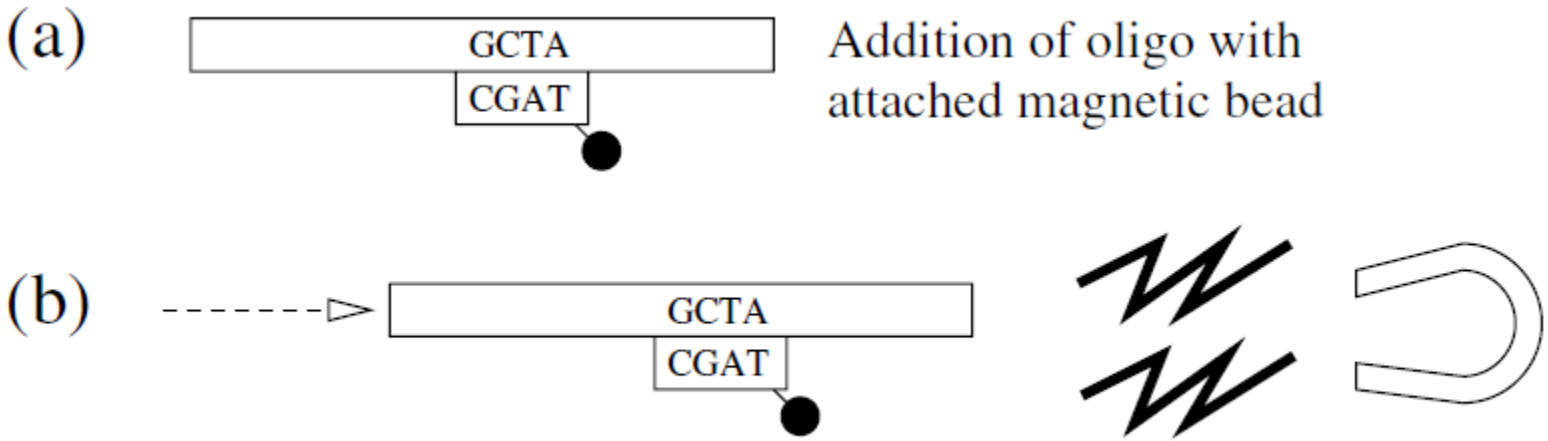
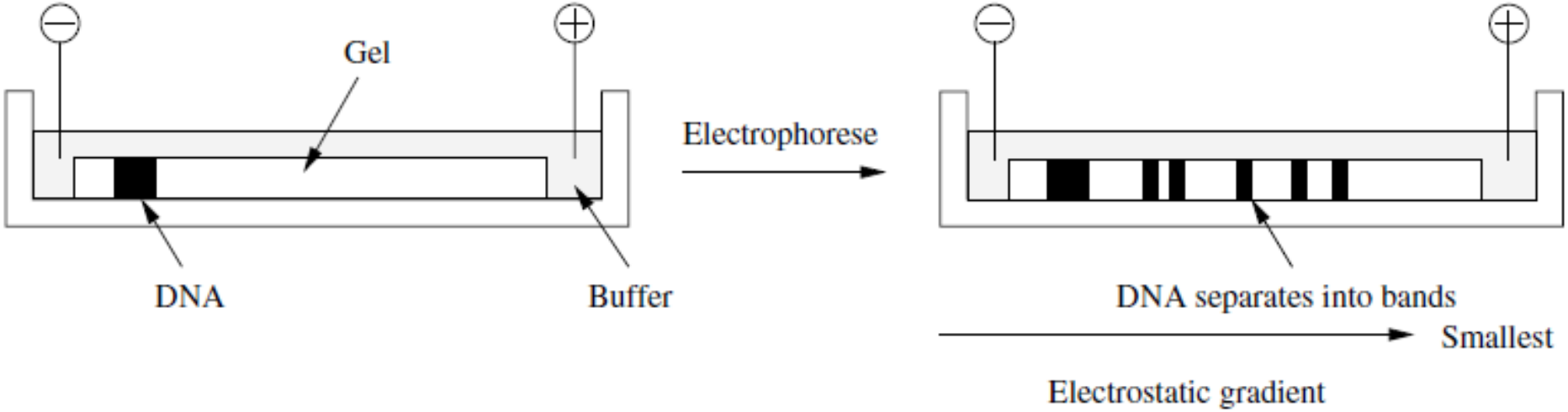


Fig. 1.8. Magnetic bead separation

Operations - 3

5. **<u>Gel electrophoresis (separation based on length)</u>**: DNA molecules have negative charge, in an electric field, they move towards the positive pole
   In a gel, the speed of movement depends on the size of the molecule



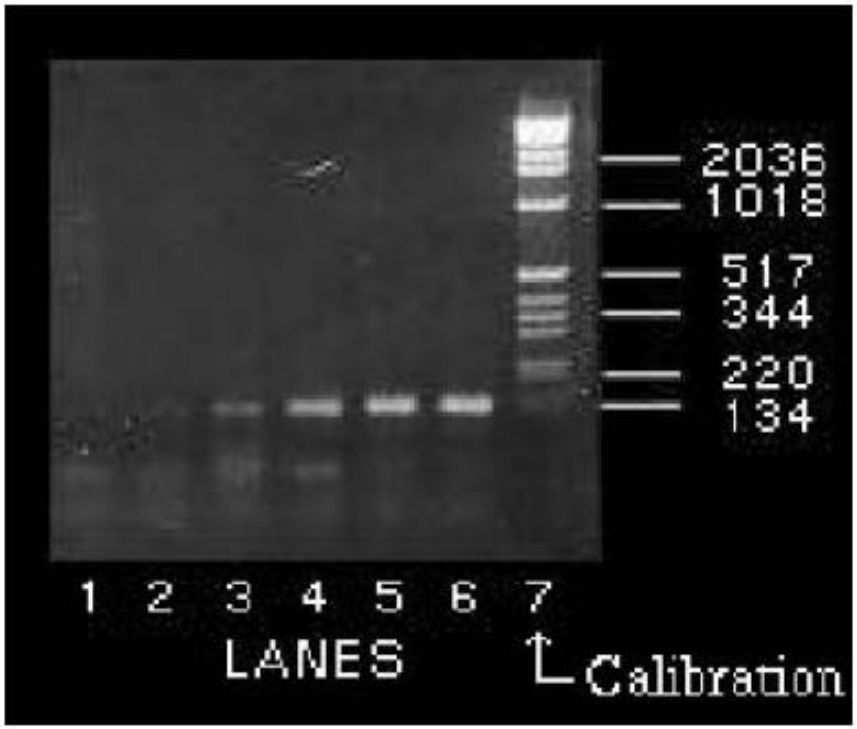


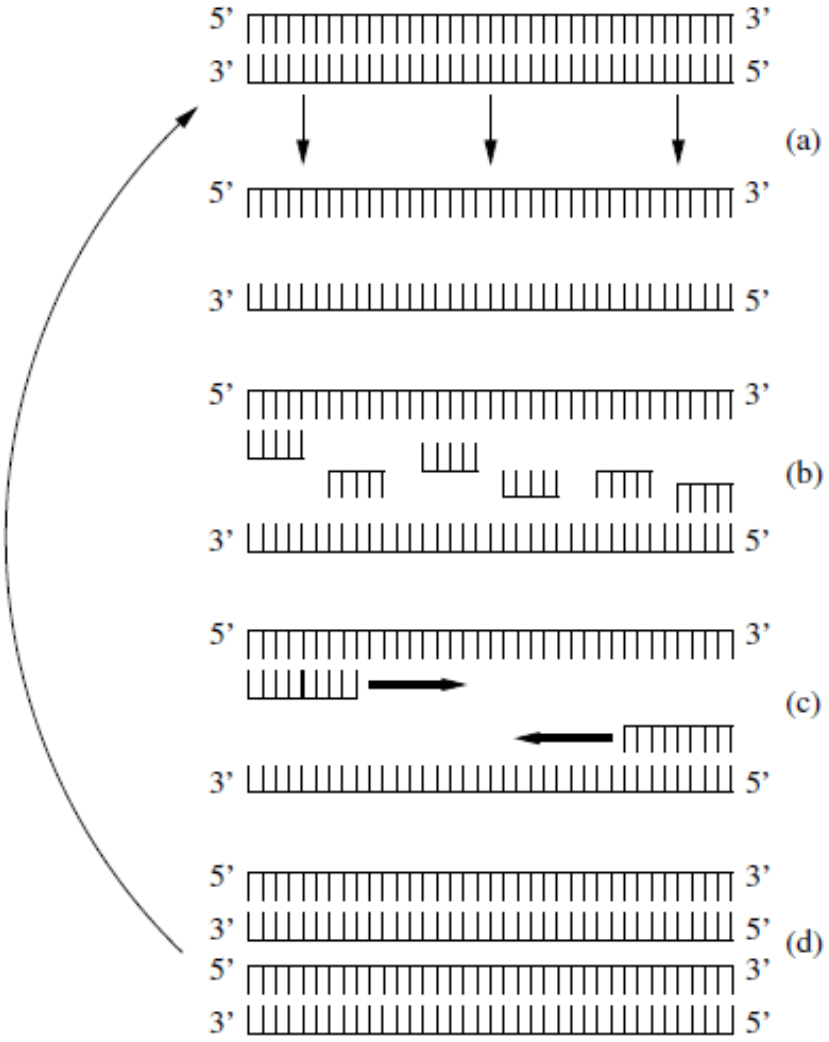
Fig. 1.10. Gel electrophoresis photograph

Operations - 4

6. **<u>Polymerase chain reaction</u>**: Polymerase enzymes can „fix” missing parts of a double string.

Fig. 1.11. (a) Primer anneals to longer template. (b) Polymerase extends primer in the 5’ to 3’ direction

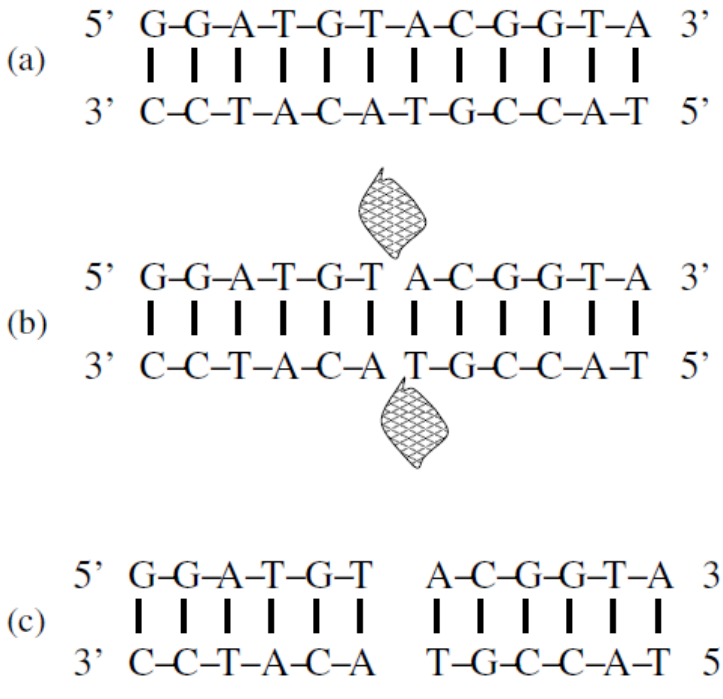
Polymerase chain reaction - multiplying molecules

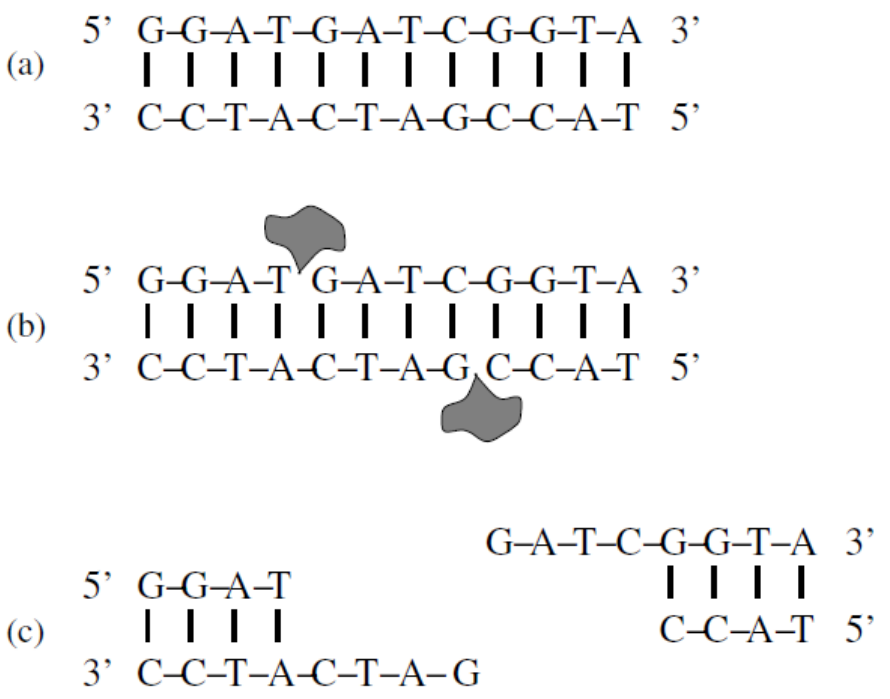


- The beginning and the end is known
- There can be upto 3000 base pairs in the „middle”.

Operations - 5

7. **<u>Cutting</u>**: Cutting double chains with enzymes (sticky ends)





ences therein.
2. M. Orrit, J. Bernard, R. I. Personov, *J. Phys. Chem.* **97**, 10256 (1993).
3. F. Güttler, T. Irngartinger, T. Plakhotnik, A. Renn, U. P. Wild, *Chem. Phys. Lett.* **217**, 393 (1994).
4. M. Ishikawa, K. Hirano, T. Hayakawa, S. Hosoi, S. Brenner, *Jpn. J. Appl. Phys.* **33**, 1571 (1994).
5. E. Betzig and R. J. Chichester, *Science* **262**, 1422 (1993).
6. J. K. Trautman, J. J. Macklin, L. E. Brus, E. Betzig, *Nature* **369**, 40 (1994).
7. D. C. Nguyen, R. A. Keller, J. H. Jett, J. C. Martin, *Anal. Chem.* **59**, 2158 (1987).
8. K. Peck, L. Stryer, A. N. Glazer, R. A. Mathies, *Proc. Natl. Acad. Sci. U.S.A.* **86**, 4087 (1989).
9. W. B. Whitten, L. M. Ramsey, S. A. Arnold, B. V. Bronk, *Anal. Chem.* **63**, 1027 (1991); K. C. Ng, W. B. Whitten, S. A. Arnold, L. M. Ramsey, *ibid.* **64**, 2914 (1992).
10. M. Eigen and R. Rigler, *Proc. Natl. Acad. Sci. U.S.A.* **91**, 5740 (1994).
11. In fluorescence correlation spectroscopy, the intensity recorded at time t is multiplied by that recorded at $t + \Delta t$, and the product is integrated over a finite period of time; see D. E. Koppel, *Phys. Rev. A* **10**, 1938 (1974).
12. T. T. Perkins, D. E. Smith, S. Chu, *Science* **264**, 819 (1994); T. T. Perkins, S. R. Quake, D. E. Smith, S. Chu, *ibid.*, p. 822.
13. S. B. Smith, L. Finzi, C. Bustamante, *ibid.* **258**, 1122 (1992); C. Bustamante, *Annu. Rev. Biophys. Biophys. Chem.* **20**, 415 (1991).
14. M. Washizu and O. Kurosawa, *IEEE Trans. Ind. Appl.* **26**, 1165 (1990).
15. S. B. Smith, P. K. Aldridge, J. B. Callis, *Science* **243**, 203 (1989).
16. N. J. Rampino and A. Chrambach, *Anal. Biochem.* **194**, 278 (1991).
17. K. Morikawa and M. Yanagida, *J. Biochem.* **89**, 693 (1981).
18. I. Auzanneau, C. Barreau, L. Salome, *C. R. Acad. Sci. Paris* **316**, 459 (1993).
19. H. Kabata *et al.*, *Science* **262**, 1561 (1993).
20. D. A. Schafer, J. Gelles, M. P. Sheetz, R. Landick, *Nature* **352**, 444 (1991).
21. Laser excitation at 488.0 and 514.5 nm was provided by an argon ion laser (Lexel Lasers, Fremont, CA). The laser beam entered the microscope through a back port and was directed to an oil-immersion objective (×100, NA = 1.3, Nikon Instrument Group, Melville, NY) by a dichroic beamsplitter (505DRLP02 or

Molecular Computation of Solutions to Combinatorial Problems

Leonard M. Adleman

The tools of molecular biology were used to solve an instance of the directed Hamiltonian path problem. A small graph was encoded in molecules of DNA, and the "operations" of the computation were performed with standard protocols and enzymes. This experiment demonstrates the feasibility of carrying out computations at the molecular level.

In 1959, Richard Feynman gave a visionary talk describing the possibility of building computers that were "sub-microscopic" (*1*). Despite remarkable progress in computer miniaturization, this goal has yet to be achieved. Here, the possibility of computing directly with molecules is explored.

A directed graph G with designated vertices v_{in} and v_{out} is said to have a Hamiltonian path (*2*) if and only if there exists a sequence of compatible "one-way" edges $e_1, e_2, \ldots, e_z$ (that is, a path) that begins at v_{in}, ends at v_{out}, and enters every other vertex exactly once. Figure 1 shows a graph that for $v_{in} = 0$ and $v_{out} = 6$ has a Hamiltonian path, given by the edges 0→1, 1→2, 2→3, 3→4, 4→5, 5→6. If the edge 2→3 were removed from the graph, then the resulting graph with the same designated vertices would not have a Hamiltonian path. Similarly, if the designated vertices were changed to $v_{in} = 3$ and $v_{out} = 5$ there would be no Hamiltonian path (because, for example, there are no edges entering vertex 0).

There are well-known algorithms for deciding whether an arbitrary directed graph with designated vertices has a Hamiltonian path or not. However, all known algorithms for this problem have exponential worst-case complexity, and hence there are instances of modest size for which these algorithms require an impractical amount of computer time to render a decision. Because the directed Hamiltonian path problem has been proven to be NP-complete, it seems likely that no efficient (that is, polynomial time) algorithm exists for solving it (*2*, *3*).

The following (nondeterministic) algorithm solves the directed Hamiltonian path problem:

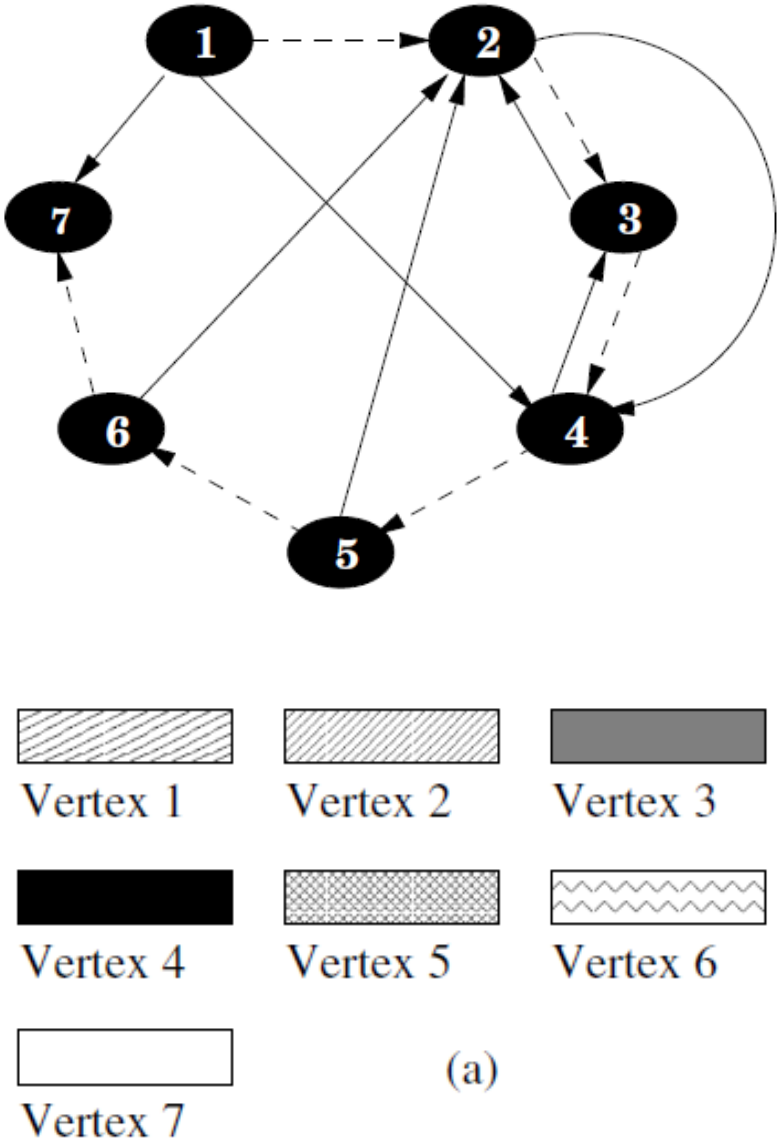
Step 1: Generate random paths through the graph.

Step 2: Keep only those paths that begin with v_{in} and end with v_{out}.

Step 3: If the graph has n vertices, then keep only those paths that enter exactly n vertices.

Step 4: Keep only those paths that enter all of

Department of Computer Science and Institute for Molecular Medicine and Technology, University of Southern California, 941 West 37th Place, Los Angeles, CA 90089, USA.



the code of a vertex is 20 base pairs long

Adleman's experiment – Phase 1

Molecules representing vertices and edges mixed in a test tube.

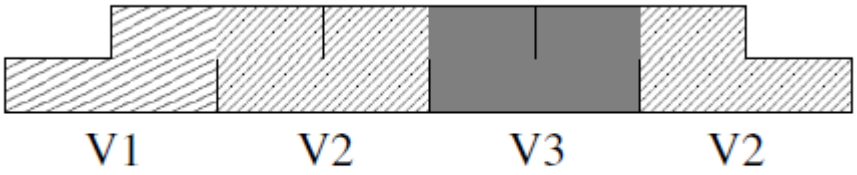
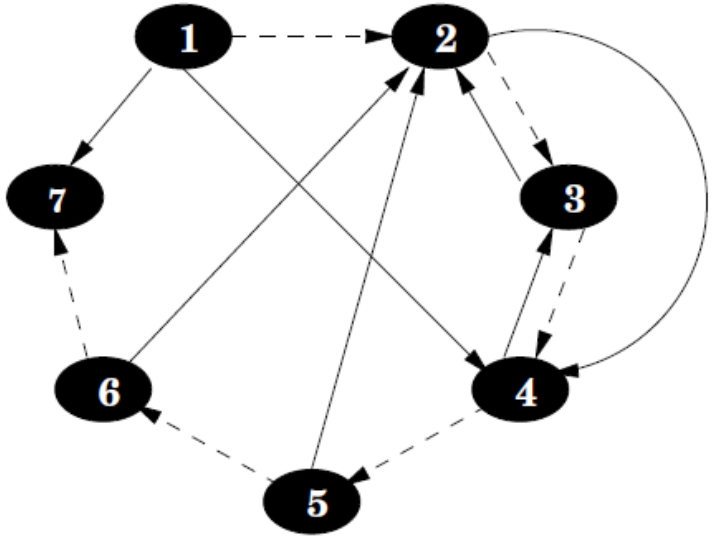


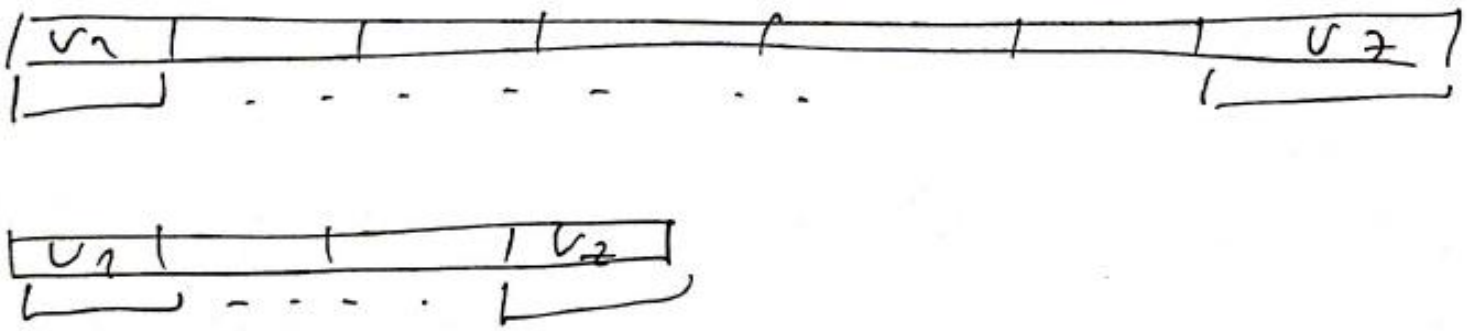
Fig. 5.4. Example path created in Adleman's scheme

Fig. 5.5. Unique Hamiltonian path



Adleman's experiment – Phase 2

Polymerase chain reaction: multiply those starting with v1, ending in v7



Choosing those which have the proper length (140 base pairs, gel electrophoresis)

Adleman's experiment – Phase 2

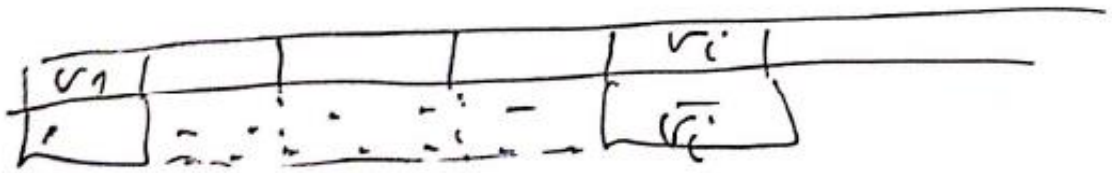
Choosing those which contain all vertices, 5 steps:

In the i-th step, we get rid of those molecules which do not contain vertex i+1

(Phase 2 removes all molecules which cannot represent solutions.)

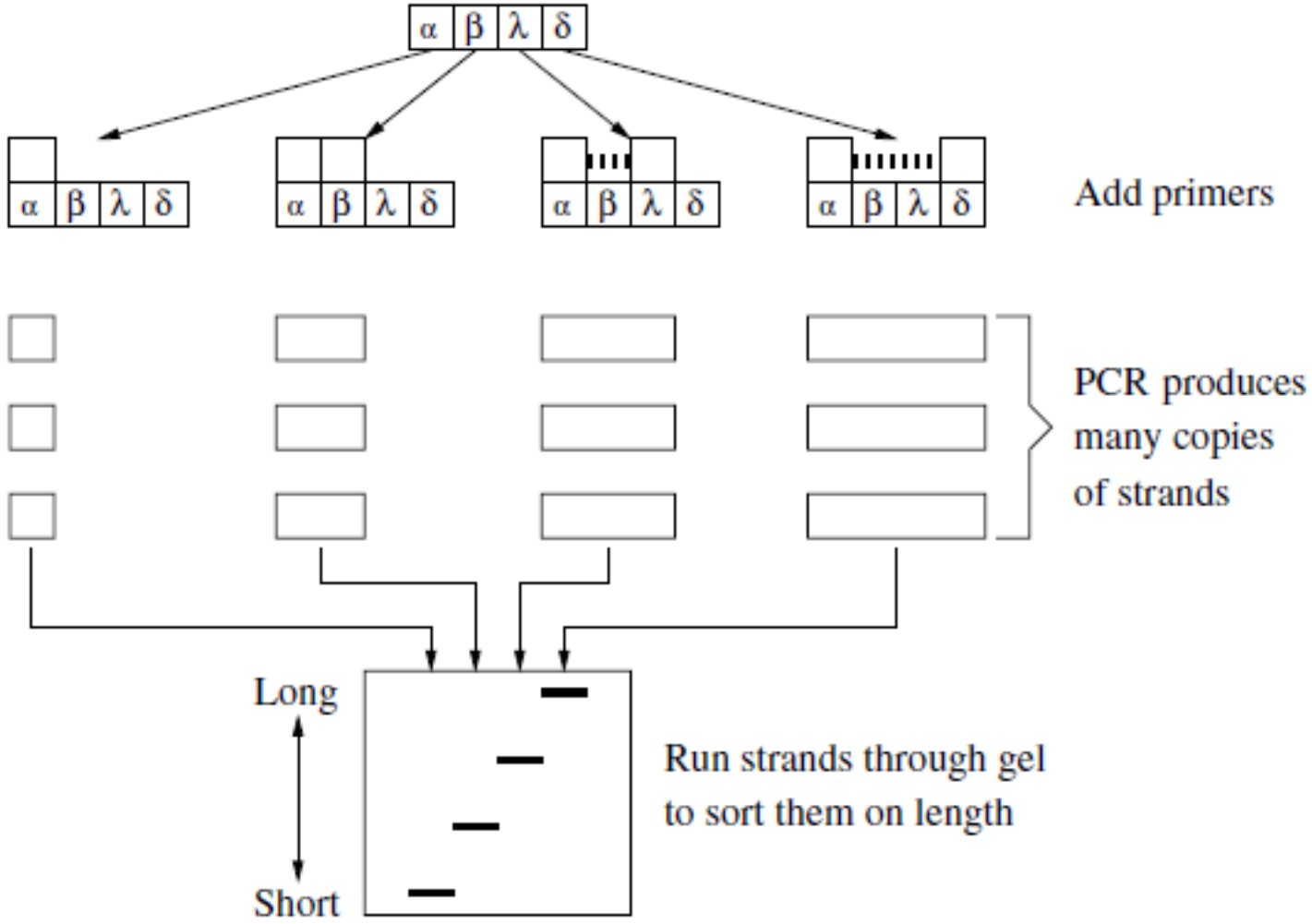
Adleman's experiment – last phase

Read off the path from the resulting molecules.



If vi is the j-the vertex in the path, then the part between v1 and vi has j*20 base pairs.

First, create the v1-vi double strand for all vi, then a series of PCR reactions…



All vertices have a path, the distance the molecules move in the path corresponds to the distance of the vertex from v1.

We can use the operations …

… to define abstract computational models

1. Filtering type algorithms
2. Constructive algorithms (self-assembly)

The abstract model

The idea (data structure): DNA molecules can be represented by strings, test tubes with solutions of DNA can be represented by string multisets.

The abstract model

Operations:

- separate(T,S): T test tube, S string, the result are the multisets +(T,S) and –(T,S), those strings of T which contain or not contain substring S.
- merge(T1,T2,…,Tn): create the union
- detect(T): returns „true” if T is not empty

(L. Adleman 1994)

Example – SAT, formula satisfiiability

Formula with n variables, for example (n=3):

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee x_3)$$

A truth assignment is represented by an n bit string.:

$$110 \Leftrightarrow x_1 = 1,\ x_2 = 1,\ x_3 = 0$$

These assignments can be represented by molecules: b1b2b3, where bi base sequence can be of two forms coding bi=1 or bi=0

A SAT algorithm (T. Lipton 1995)

(1) create T
(2) for each clause c
(3) for each literal v
(4) if v is „positive”, collect those assignements from T, which assign 1 to v, otherwise collect those, which assign 0 to v
(5) end
(6) merge the collected assignments in a new T
(7) end
(8) if T is not empty, the formula is satisfiable

(1) Input(T)
(2) **for** $a = 1$ to $|I|$ **do begin**
(3) **for** $b = 1$ to $|C_a|$ **do begin**
(4) **if** $v_b^a = x_j$ **then** $T_b \leftarrow +(T, v_b^a = 1)$
else $T_b \leftarrow +(T, v_b^a = 0)$
(5) **end for**
(6) $T \leftarrow merge(T_1, T_2, \ldots, T_{|C_a|})$
(7) **end for**
(8) Output(detect(T))

(polynomial time complexity)

Filtering type algorithms in general

1. Create DNA molecules representing **all possible solution candidates**
2. **Remove** those, which cannot be solutions
3. If something **remains**, it is the solution

For example: L. Adleman’s experiment, or the previous SAT algorithm

Other experiments

Molecular computation: RNA solutions to chess problems

Dirk Faulhammer*, Anthony R. Cukras*, Richard J. Lipton†, and Laura F. Landweber*‡

Departments of *Ecology and Evolutionary Biology and †Computer Science, Princeton University, Princeton, NJ 08544

Communicated by Andrew Chi-Chih Yao, Princeton University, Princeton, NJ, December 3, 1999 (received for review April 25, 1999)

We have expanded the field of "DNA computers" to RNA and present a general approach for the solution of satisfiability problems. As an example, we consider a variant of the "Knight problem," which asks generally what configurations of knights can one place on an $n \times n$ chess board such that no knight is attacking any other knight on the board. Using specific ribonuclease digestion to manipulate strands of a 10-bit binary RNA library, we developed a molecular algorithm and applied it to a 3 × 3 chessboard as a 9-bit instance of this problem. Here, the nine spaces on the board correspond to nine "bits" or placeholders in a combinatorial RNA library. We recovered a set of "winning" molecules that describe solutions to this problem.

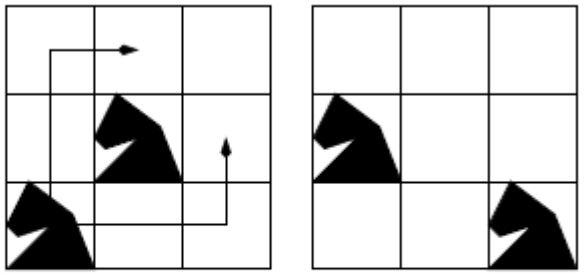
DNA computing | satisfiability | RNA evolution | *in vitro* selection | SELEX

(8). Annealing of the four half-sized strands i
containing 1 μM of each of four halves (
extension during five thermocycles (94°C for
and 72°C for 30 sec per cycle) generated th
sized strands with a total of 1,024 (2^{10}) unic
the 3 × 3 instance of this problem considers
512 different chessboards. Bit 10 was unus
tions.) The degeneracy of the DNA pool
sequencing (10) using 5′ end-radiolabeled
$[CAT]_4$CTCGAGAATT) and 32.41 (
$[CTA]_4$CGGGATCCTAATGACCAAGG)
reactions (SequiTherm Excel, Epicentre Te

The 271-bp DNA pool was amplified
T7TXR (TTCTAATACGACTCACTA
GAGAATT) and 32.41 (1 μM each: 15 cy

[Faulhammer et al. 2000]

- Knights on a chess board (A SAT variant)



(a) (b)

(a) Legal configuration. (b) Illegal configuration -

a	b	c
d	e	f
g	h	i

(the variable for a position is 1 if there is a knight there, 0 otherwise)

$$((\neg h \wedge \neg f) \vee \neg a) \wedge ((\neg g \wedge \neg i) \vee \neg b) \wedge ((\neg d \wedge \neg h) \vee \neg c) \wedge ((\neg c \wedge \neg i) \vee \neg d) \wedge ((\neg a \wedge \neg g) \vee \neg f) \wedge ((\neg b \vee \neg f) \vee \neg g) \wedge ((\neg a \wedge \neg c) \vee \neg h) \wedge ((\neg d \wedge \neg b) \vee \neg i)$$

The algorithm

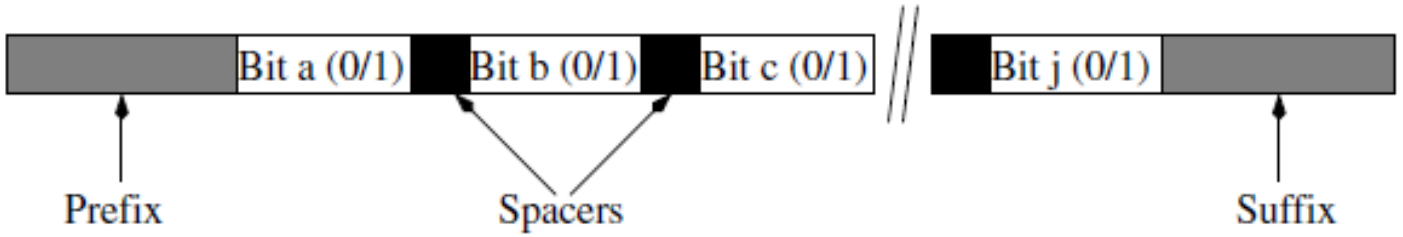
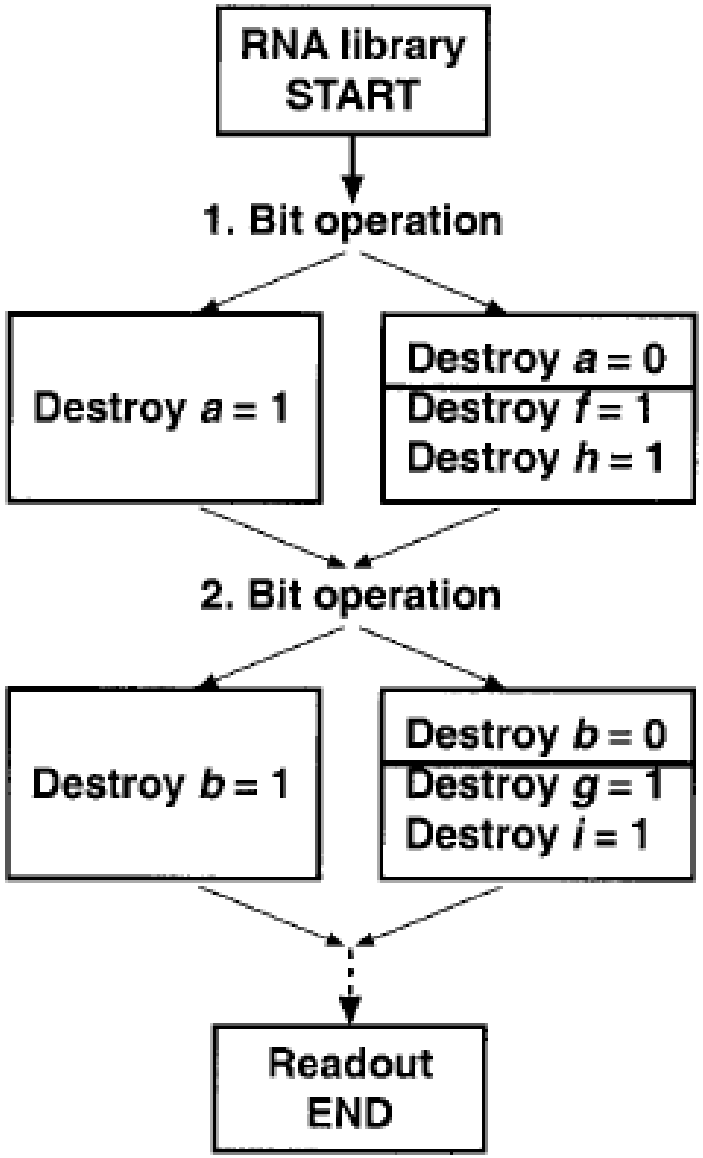


Fig. 5.15. Template for RNA strands

1. For each square, sequentially, split the RNA library into two tubes, labelled 1 and 2. After digestions have taken place, tube 1 will contain strands that contain a knight at that square, and tube 2 will contain strands that do *not* have knights at that square
2. In tube 1, digest with RNase H strands that have no knight at position **a**, as well as strands that describe a knight at attacking positions **h** and **f**. This implements the logical statement $((\neg h \wedge \neg f) \vee \neg a)$
3. In tube 2, digest strands that have a knight present at position **a**
4. Remove the DNA oligos used to perform the above digestions
5. Go to step 1, repeating with square **b**


RNA library
START
1. Bit operation
Destroy a = 1
Destroy a = 0
Destroy f = 1
Destroy h = 1
2. Bit operation
Destroy b = 1
Destroy b = 0
Destroy g = 1
Destroy i = 1
Readout
END

The result

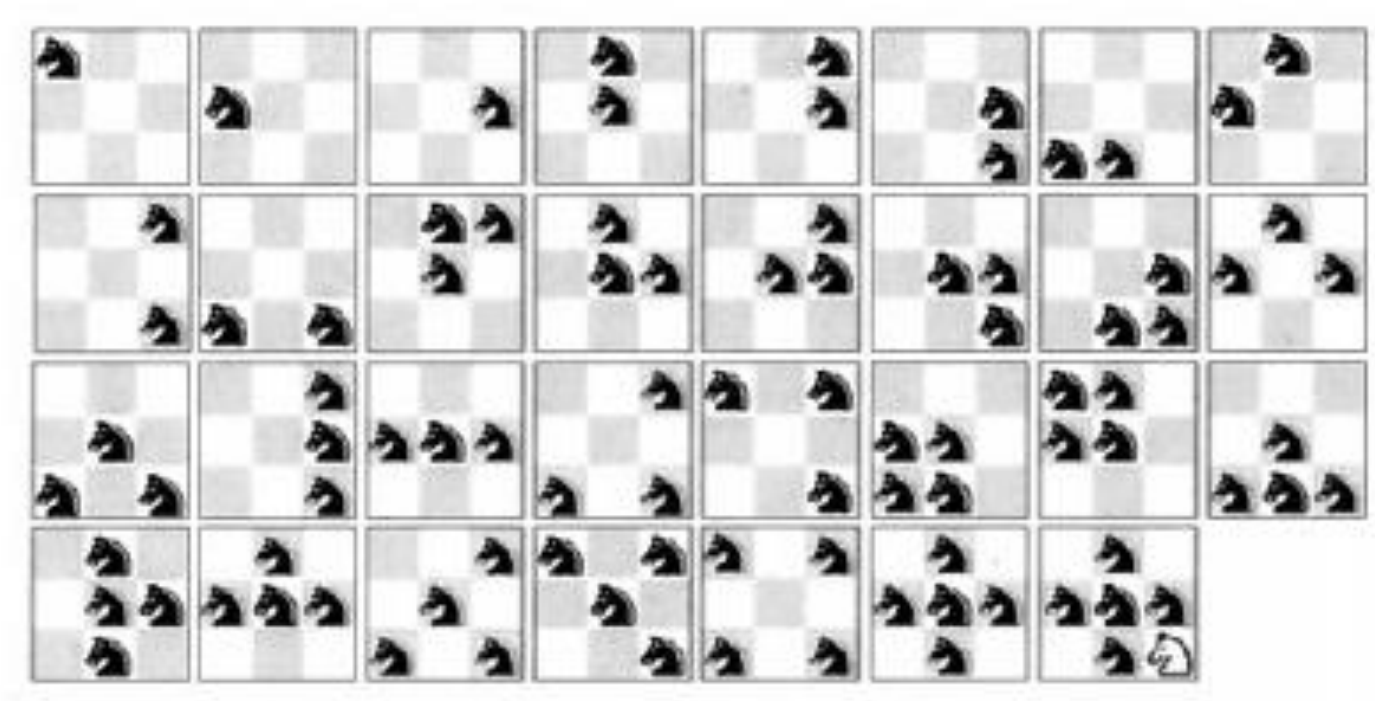


Fig. 4. Representations of the 31 unique boards analyzed by PCR readout. The last board contains one "illegal" white knight.

An other SAT solution

25. Weis, R. M. & McConnell, H. M. Cholesterol stabilizes the crystal-liquid interface in phospholipid monolayers. *J. Phys. Chem.* **89,** 4453–4459 (1985).
26. Chunbo, Y. *et al.* Lanthanide ion induced formation of stripes domain structure in phospholipid Langmuir-Blodgett monolayers film observed by atomic force microscopy. *Surf. Sci.* **366,** L729–L734 (1996).
27. Biebuyck, H. A. & Whitesides, G. M. Self-organisation of organic liquids on patterned self-assembled monolayers of alkanthiols on gold. *Langmuir* **10,** 2790–2793 (1994).
28. Kim, E., Xia, Y. & Whitesides, G. M. Polymer microstructures formed by moulding in capillaries. *Nature* **376,** 581–584 (1995).
29. Xia, Y. & Whitesides, G. M. Soft Lithography. *Annu. Rev. Mater. Sci.* **28,** 153–184 (1998).
30. Anczykowski, B., Gotsmann, B., Fuchs, H., Cleveland, J. P. & Elings, V. B. How to measure energy dissipation in dynamic mode atomic force microscopy. *Appl. Surf. Sci.* **140,** 376–382 (1999).

Acknowledgements

We thank G. Schmid for providing the Au_{55} clusters. This work was supported by the Deutsche Forschungsgemeinschaft.

Correspondence and requests for materials should be addressed to L.F.C. (e-mail:chi@nwz.uni-muenster.de).

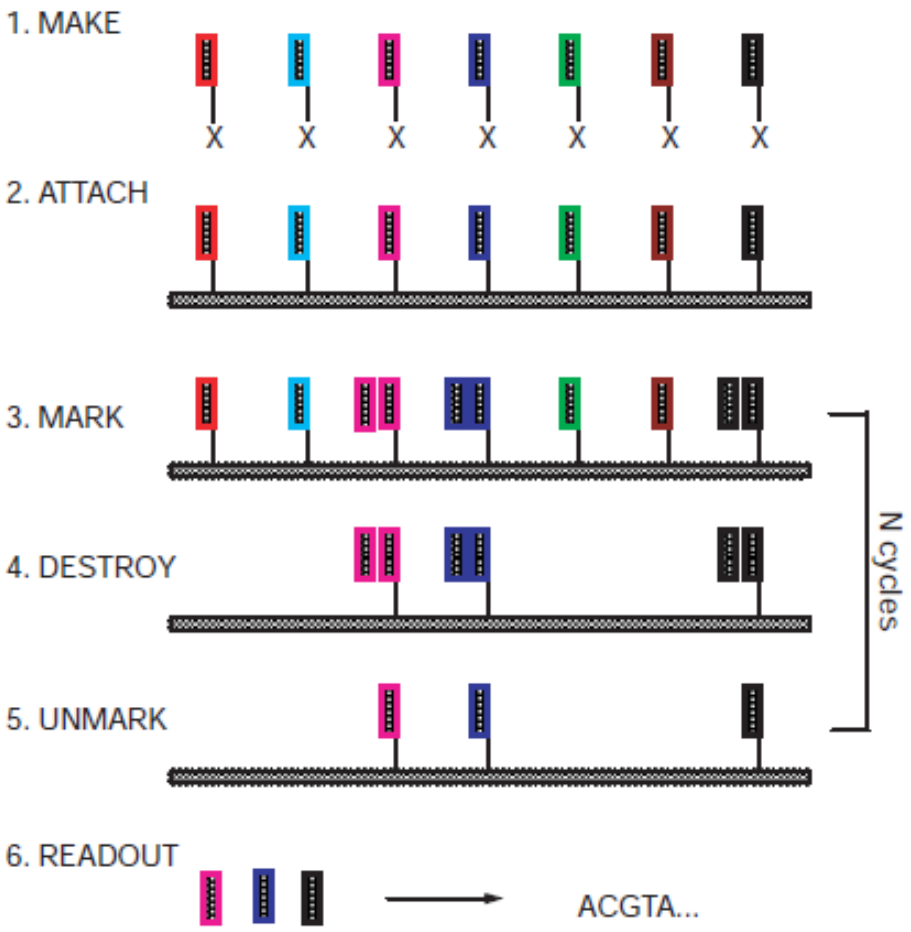
DNA computing on surfaces

Qinghua Liu*†, Liman Wang*, Anthony G. Frutos*†, Anne E. Condon†‡, Robert M. Corn* & Lloyd M. Smith*

** Department of Chemistry, ‡ Department of Computer Science, University of Wisconsin, Madison, Wisconsin 53706, USA*

$$(w \vee x \vee y) \wedge (w \vee \bar{y} \vee z) \wedge (\bar{x} \vee y) \wedge (\bar{w} \vee \bar{y})$$

Strand	*Sequence*	*wxyz*
S_0	CAACCCAA	0000
S_1	TCTCAGAG	0001
S_2	GAAGGCAT	0010
S_3	AGGAATGC	0011
S_4	ATCGAGCT	0100
S_5	TTGGACCA	0101
S_6	ACCATTGG	0110
S_7	GTTGGGTT	0111
S_8	CCAAGTTG	1000
S_9	CAGTTGAC	1001
S_10	TGGTTTGG	1010
S_11	GATCCGAT	1011
S_12	ATATCGCG	1100
S_13	GGTTCAAC	1101
S_14	AACCTGGT	1110
S_15	ACTGGTCA	1111



A „gel based” computer

Solution of a 20-Variable 3-SAT Problem on a DNA Computer

Ravinderjit S. Braich,[1] Nickolas Chelyapov,[1] Cliff Johnson,[1]
Paul W. K. Rothemund,[2] Leonard Adleman[1]*

A 20-variable instance of the NP-complete three-satisfiability (3-SAT) problem was solved on a simple DNA computer. The unique answer was found after an exhaustive search of more than 1 million (2^{20}) possibilities. This computational problem may be the largest yet solved by nonelectronic means. Problems of this size appear to be beyond the normal range of unaided human computation.

The vast parallelism, exceptional energy efficiency, and extraordinary information den-

oligonucleotide probes immobilized in polyacrylamide gel–filled glass modules. Infor-

Th
comp
3-con
Φ (F
challe
have
(Fig.
an ite
seen
note,
under
and i
(1,04
proce
signm
that a
la wo

The formula

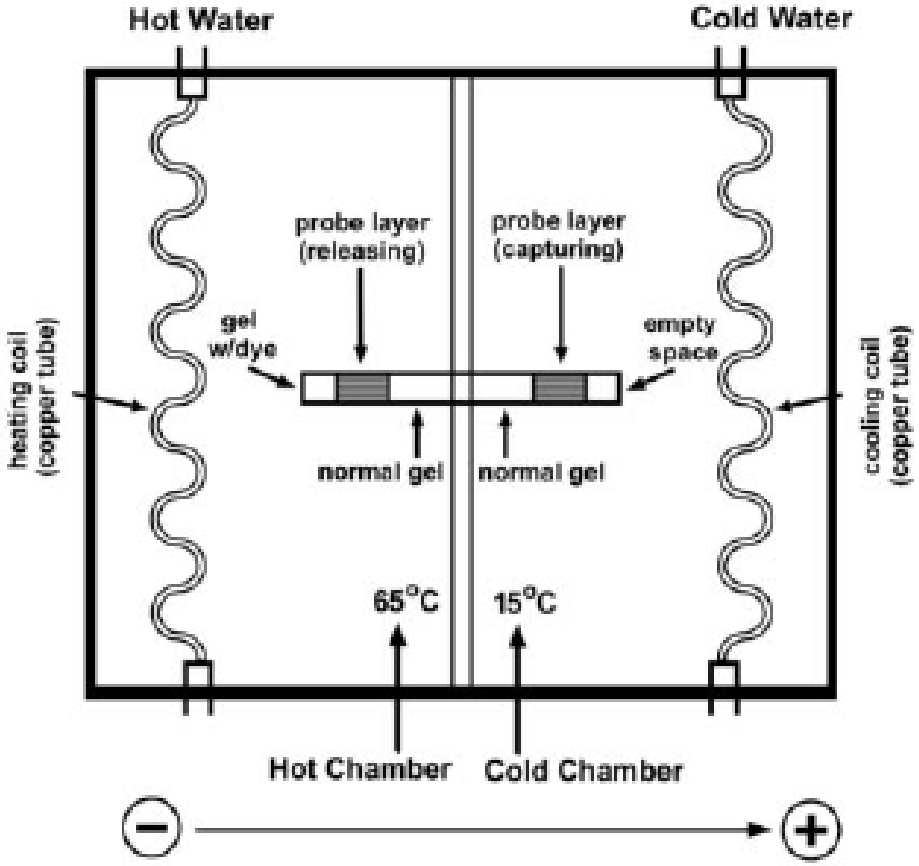
$$\Phi = (\neg x_{13} \vee x_{16} \vee x_{18}) \wedge (x_5 \vee x_{12} \vee \neg x_9) \wedge (\neg x_{13} \vee \neg x_2 \vee x_{20}) \wedge (x_{12} \vee x_9 \vee \neg x_5) \wedge (x_{19} \vee \neg x_4 \vee x_6) \wedge (x_9 \vee x_{12} \vee \neg x_5) \wedge (\neg x_1 \vee x_4 \vee \neg x_{11}) \wedge (x_{13} \vee \neg x_2 \vee \neg x_{19}) \wedge (x_5 \vee x_{17} \vee x_9) \wedge (x_{15} \vee x_9 \vee \neg x_{17}) \wedge (\neg x_5 \vee \neg x_9 \vee \neg x_{12}) \wedge (x_6 \vee x_{11} \vee x_4) \wedge (\neg x_{15} \vee \neg x_{17} \vee x_7) \wedge (\neg x_6 \vee x_{19} \vee x_{13}) \wedge (\neg x_{12} \vee \neg x_9 \vee x_5) \wedge (x_{12} \vee x_1 \vee x_{14}) \wedge (x_{20} \vee x_3 \vee x_2) \wedge (x_{10} \vee \neg x_7 \vee \neg x_8) \wedge (\neg x_5 \vee x_9 \vee \neg x_{12}) \wedge (x_{18} \vee \neg x_{20} \vee x_3) \wedge (\neg x_{10} \vee \neg x_{18} \vee \neg x_{16}) \wedge (x_1 \vee \neg x_{11} \vee \neg x_{14}) \wedge (x_8 \vee \neg x_7 \vee \neg x_{15}) \wedge (\neg x_8 \vee x_{16} \vee \neg x_{10})$$

with a unique satisfying assignment of

$$x_1 = F, x_2 = T, x_3 = F, x_4 = F, x_5 = F, x_6 = F, x_7 = T, x_8 = T, x_9 = F, x_{10} = T, x_{11} = T, x_{12} = T, x_{13} = F, x_{14} = F, x_{15} = T, x_{16} = T, x_{17} = T, x_{18} = F, x_{19} = F, x_{20} = F.$$

The computation proceeds as follows: for each clause, a glass clause module is constructed which is filled with gel and contains covalently bound probes designed to capture only those library strands that *do satisfy* that clause; strands that do not satisfy the clause are discarded.

The machinery



Next

- Limitations of filtering type models
- Different approaches:
  - Self-assembly

EATCS Bulletin Feb. 95 (no. 55) pp. 136-138

The Structural Complexity Column

by

Juris HARTMANIS

Cornell University, Department of Computer Science
Ithaca, NY 14853, U.S.A.
jh@cs.cornell.edu

On the Weight of Computations

One of the Grand Challenges to computer science is to understand what is and is not feasibly computable. Recursive function theory clarified what is and is not effectively computable and in the process extended our understanding of Goedel incompleteness results about the limits of the power of formal mathematical methods. Since computing is universal and encompasses the power of mathematics, the understanding of the limits of the feasibly computable could give a deeper

The argument of Hartmanis

- The exponential function seems to give an upper limit for the feasibly computable (time, memory, weight requirements should not grow exponentially with the size of the problem).

The argument of Hartmanis

The following nondeterministic algorithm was used to solve the directed Hamiltonian path problem:

Step 1: Generate random paths through the graph.

Step 2: Keep only paths that begin with in-node and end with out-node.

Step 3: If the graph has n vertices, then keep only those paths that enter exactly n vertices.

Step 4: Keep only those paths that enter all of the vertices of the graph at least once.

Step 5: If any paths remain, say "Yes'; otherwise, say "No".

The argument of Hartmanis

- At step 1, almost all possible paths have to be generated, for the algorithm to work.

- Each molecule has a certain weight.

Let us calculate how heavy the initial test tubes may get.

The argument of Hartmanis

- Consider a graph with 200 nodes
- 200 nodes have to be coded with a 4 letter alphabet, the length of these coding sequences cannot be less than $\log_4 200$
- A base weighs 10^{-25} kg

Thus, the initial set weighs:

$$2^{200} \cdot \log_4 200 \cdot 10^{-25}\,\text{kg} \geq (2^4)^{50} \cdot 3 \cdot 10^{-25}\,\text{kg} \geq 3 \cdot 10^{25}\,\text{kg}$$

which is more than the weight of the planet Earth.

- **Limits of the filtering type models**
  - improvements to the filtering models,
  - invention of more sophisticated models

- Different approaches (basic ideas and simple examples):
  - Self-assembly

- Limits of the filtering type models

  - improvements to the filtering models
  - invention of more sophisticated models

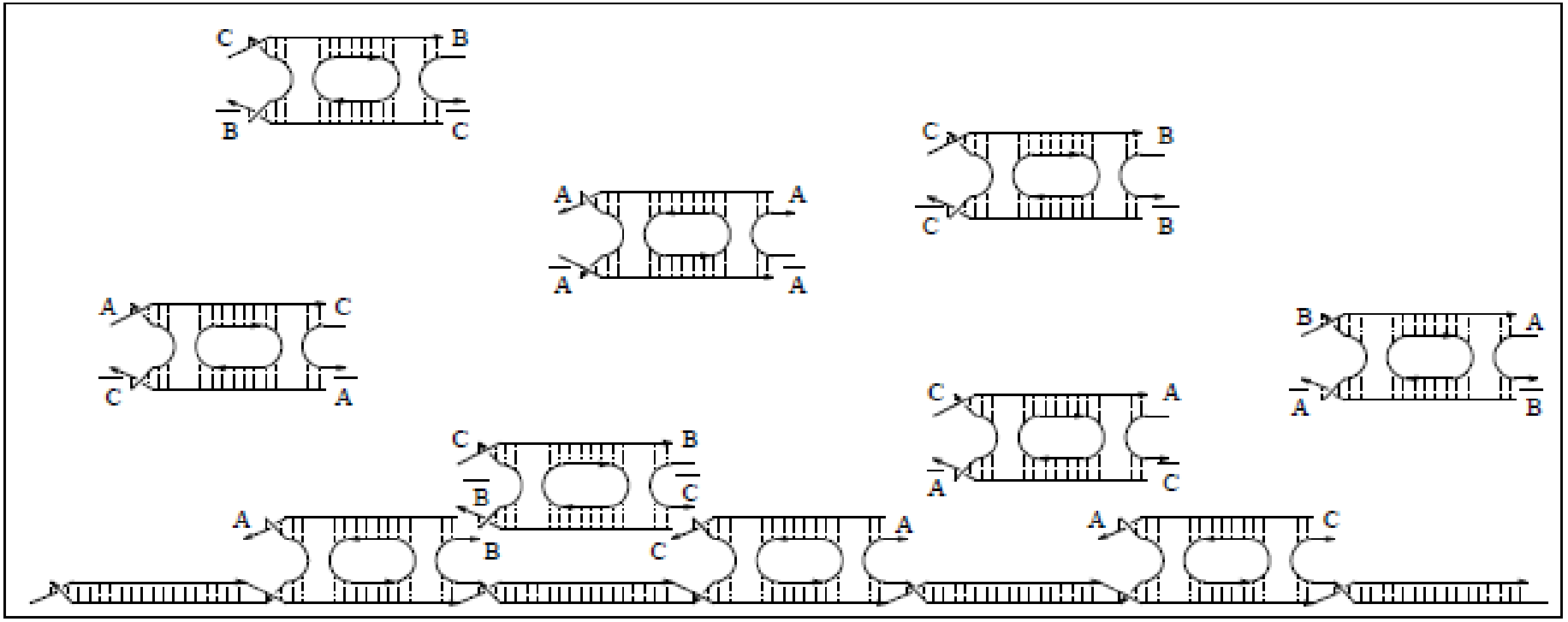
- **Different approaches (basic ideas and simple examples):**

  - **Self-assembly**

Self-assembly

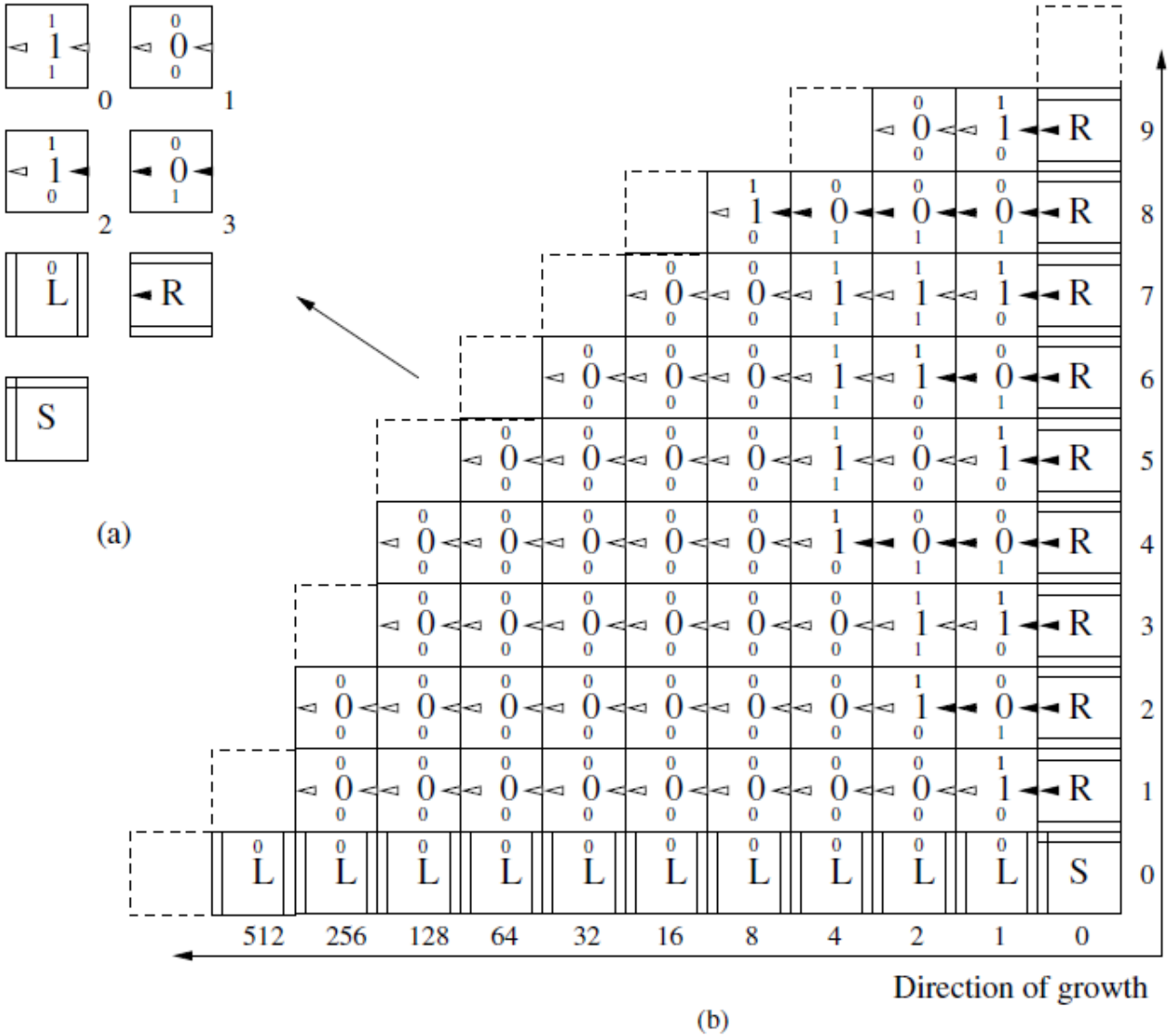
Two dimensional building blocks out of DNA molecules:

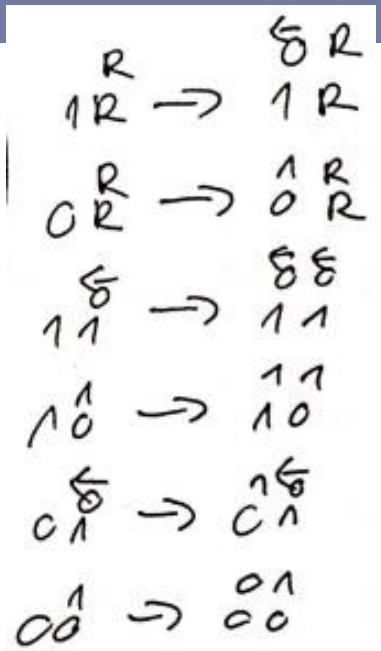
Figure 4. Rule table molecules assemble into the lattice.



E. Winfree, 1995

An example – binary counter

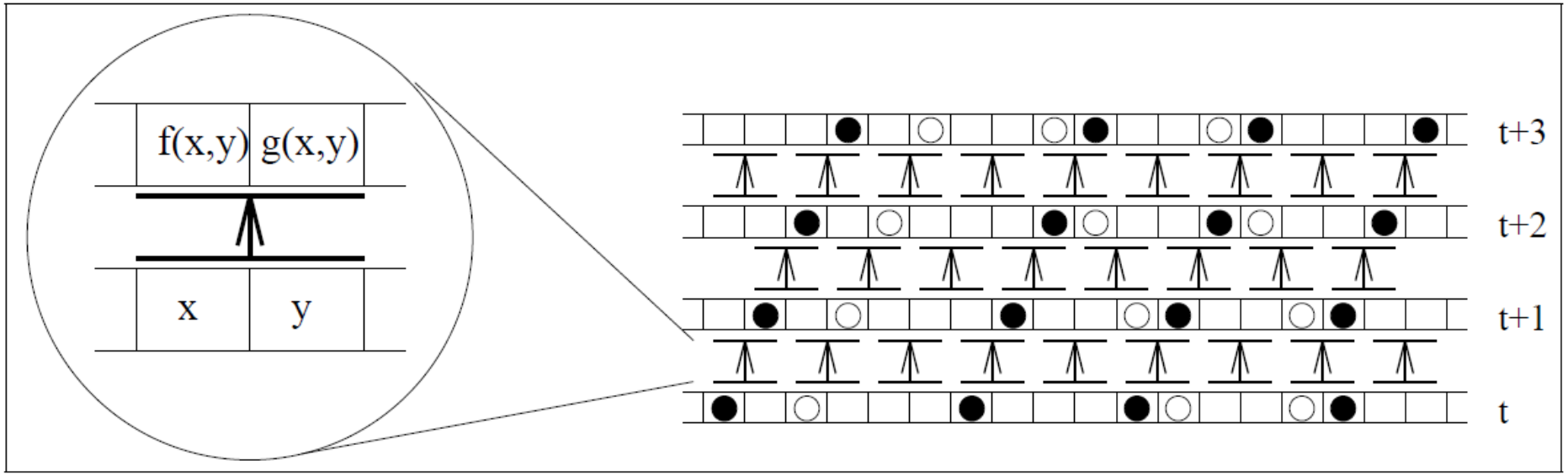




How can we compute?

For example: **Blocked cellular automata**, simple variants are equivalent with the Turing machine

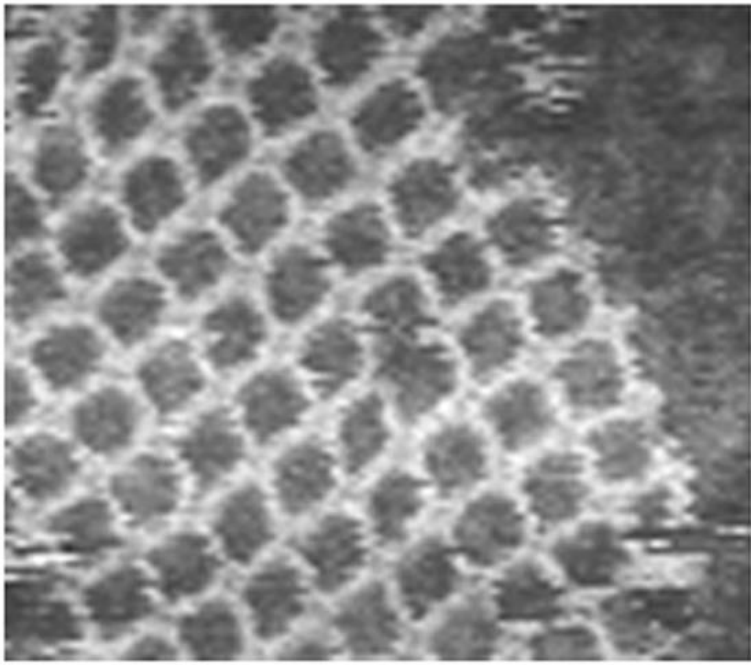
Figure 2. Operation of a blocked cellular automaton.

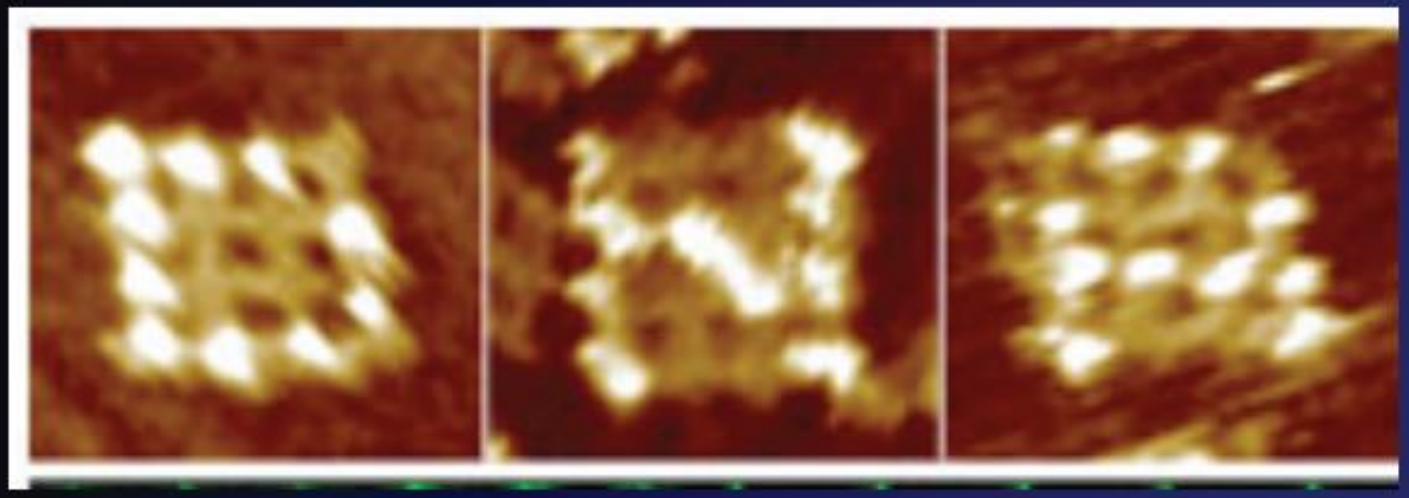


Even more interesting – creating nano-structures

A

B





Sung Ha Park, Constantin Pistol, Sang Jung Ahn, John H. Reif, Alvin R. Lebeck, Chris Dwyer, and Thomas H. LaBean, Finite-Size, Fully Addressable DNA Tile Lattices Formed by Hierarchical Assembly Procedures, Angewandte Chemie [International Edition], pp. 735-739, Volume 45, Issue 5, January 23, 2006.