Malenovice, May 2-5, 2013.

Classical logic axioms in DNA computing

Łukasz Rogowski [1, 2]

(rogowski@math.uni.lodz.pl)

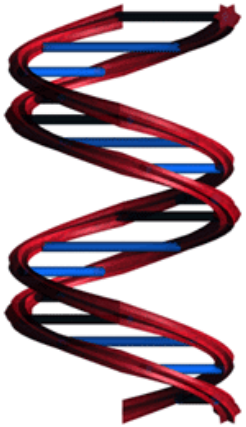
[1] University of Lodz, POLAND,
[2] Silesian University in Opava, CZECH REPUBLIC

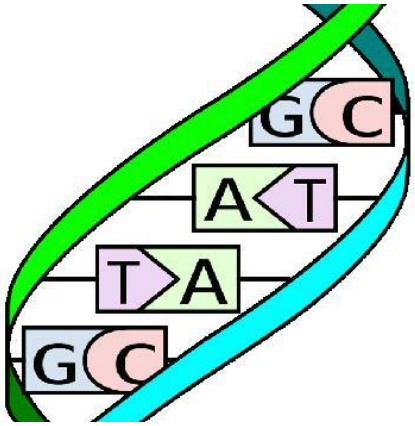
DNA Computing

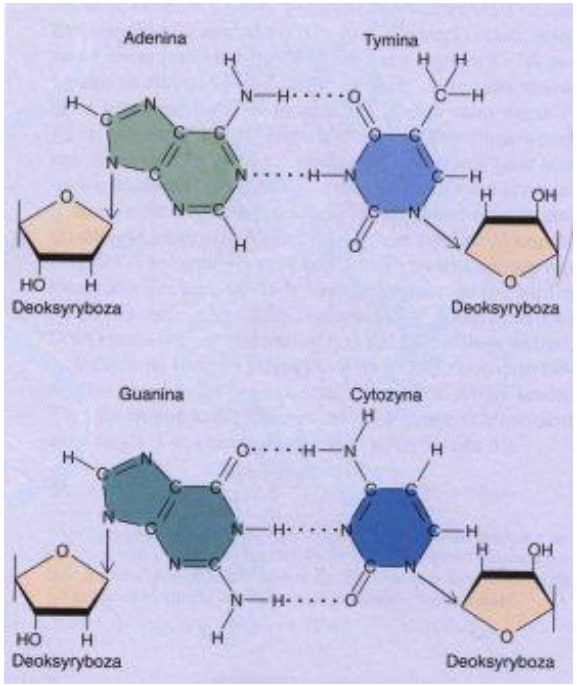
- new computational paradigm,
- alternative to traditional computers,
- use of biological molecules,
- massive parallelism of reactions,
- the fastest and the smallest,
- data storage compatibility with live organisms.

DNA foundations

- deoxyribonucleic acid,
- double-stranded sequences,
- adenine (A), guanine (G), cytosine (C), thymine (T),
- Watson-Crick complementarity (A – T), (C – G).

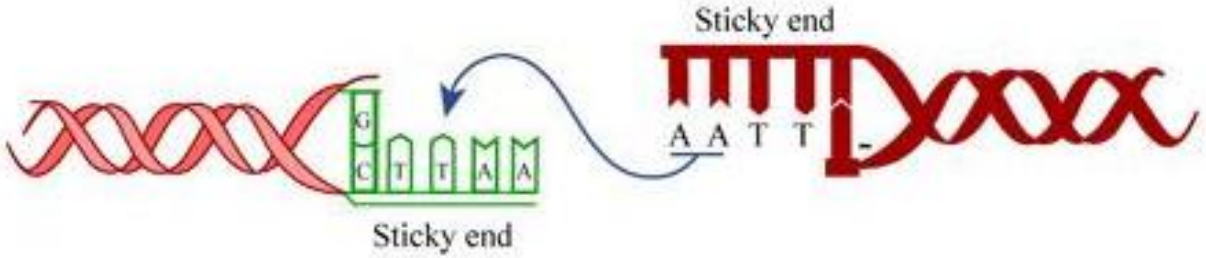




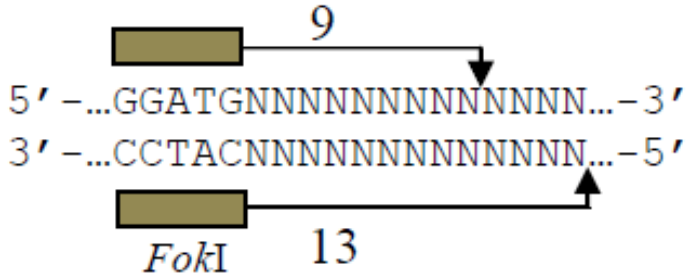


DNA operations

- ligation of complementary chains,



- cutting molecules by restriction enzymes,



- answer detecting by gel electrophoresis.

First experiment

- by Leonard Adleman in 1994,
- solving the Hamiltonian problem,
- only ligations are used,
- every possible paths are created,
- positive laboratory tests.

Subsequent works:

- other NP-hard problems, SAT problem,
- implementations of finite automatons,
- implementations of logic gates, logical inference,
- basic ideas of mathematical operations.

My scientific work

- developing of logical inference system,
- both term's values: true and false,
- using mathematical axioms,
- splicing system (alternately ligations and restrictations).

Inspirations:

- O. Unold, M. Troć „Biomolecular models for logic computable”, Inżynieria Komputerowa (2005),
- E. Shapiro, S. Kaplan, T. Ran „Molecular implementation of simple logic programs”, Nature Nanotechnology (2009).

Mathematical axioms

- $\sim(\sim p) \equiv p$ - rule of double negation,
- $(p \wedge (p \rightarrow q)) \rightarrow q$ - conditional elimination, *modus ponens*,
- $(p \rightarrow q) \equiv ((\sim q) \rightarrow \sim p)$ - rule of contraposition, *modus tullens*,
- $(\sim(p \wedge q)) \equiv ((\sim p) \vee \sim q)$ - first de Morgan's law,
- $(\sim(p \vee q)) \equiv ((\sim p) \wedge \sim q)$ - second de Morgan's law,
- $(p \vee q) \equiv ((\sim p) \rightarrow q)$ - disjunction by negation and conditional,
- $(p \equiv q) \equiv ((p \rightarrow q) \wedge (q \rightarrow p))$ - biconditional by conditionals,
- $((p \vee q) \rightarrow r) \equiv ((p \rightarrow r) \wedge (q \rightarrow r))$ - disjunction in antecedent,
- $(p \rightarrow (q \wedge r)) \equiv ((p \rightarrow q) \wedge (p \rightarrow r))$ - conjunction in consequent.

Identifying propositions

- restriction enzyme Bse*XI*,
- sticky ends with 4-nucleotides length,
- facts identified by unique sequence,
- negation identified by complementary sequence.

- example:
  a: 5'-**AATC**-3', ~***a***: 5'-**GATT**-3'.

- simple representation:
  a: 5'-**(a)**-3', ~***a***: 5'-**(~a)**-3'.

Terms representation

- representation of fact ***a***:

```
5'-(50)GCAGCGCTG     -3'    5'-( a)CAGCGCTGC(50)-3'
3'-(50)CGTCGCGAC( a)-5'     3'-    GTCGCGACG(50)-5'
```

- representation of question about fact ***a***:

```
5'-(~a)ATAT(250)-3'         5'-(250)ATAT    -3'
3'-    TATA(250)-5'         3'-(250)TATA(~a)-5'
```

Conditional

- implication ***a*⇒*b***:

```
5'-(~a)CAGCGCTG     -3'
3'-    GTCGCGAC( b)-5'
```

- which also means ~***a*⇒~*b***:

```
5'-( b)CAGCGCTG     -3'
3'-    GTCGCGAC(~a)-5'
```

- terminal molecule for every conditional:

```
5'-(50)G      -3'
3'-(50)CGTCG-5'
```

De Morgan's laws

- $(a \wedge b) \Rightarrow c$ means also $\sim c \Rightarrow (\sim a \vee \sim b)$:

```
5'-(~a)CAGCGCTG( b)(~b)CAGCGCTG     -3'
3'-    GTCGCGAC(~b)( b)GTCGCGAC( c)-5'
```

- $(a \vee b) \Rightarrow c$ means also $\sim c \Rightarrow (\sim a \wedge \sim b)$:

```
5'-(~a)CAGCGCTG     -3'    5'-(~b)CAGCGCTG     -3'
3'-    GTCGCGAC( c)-5'    3'-    GTCGCGAC( c)-5'
```

Reaction: *a*, *b*⇒~*a*, ~*b*?

- elements of reaction:

```
5'-(50)GCAGCGCTG      -3'    5'-(50)G      -3'
3'-(50)CGTCGCGAC( a)-5'      3'-(50)CGTCG-5'

5'-(~b)CAGCGCTG       -3'    5'-( b)ATAT(250)-3'
3'-    GTCGCGAC(~a)-5'       3'-    TATA(250)-5'
```

- ligation between ***a*** and ***b*⇒~*a***:

```
5'-(50)GCAGCGCTG(~a)CAGCGCTG      -3'
3'-(50)CGTCGCGAC( a)GTCGCGAC(~b)-5'
```

Reaction: *a*, *b*⇒~*a*, ~*b*?

- cutting by restriction enzyme Bse*XI*:

```
5'-(50)GCAGCGCTG(~a)      -3'    5'-CAGCGCTG      -3'
3'-(50)CGTCGCGAC( a)GTCG-5'      3'-     CGAC(~b)-5'
```

- ligation between the result and terminal molecule:

```
5'-(50)GCAGCGCTG     -3'
3'-(50)CGTCGCGAC(~b)-5'
```

- ligation between ***b*** and ***b?***:

```
5'-(50)GCAGCGCTG( b)ATAT(250)-3'
3'-(50)CGTCGCGAC(~b)TATA(250)-5'
```

Reaction: *a*, *b*⇒~*a*, ~*b*?

- cutting by restriction enzyme Bse*XI*:

```
5'-(50)GCAGCGCTG( b)      -3'    5'-ATAT(250)-3'
3'-(50)CGTCGCGAC(~b)TATA-5'      3'-      (250)-5'
```

- ligation between two representations of the result:

```
5'-(250)ATAT(250)-3'
3'-(250)TATA(250)-5'
```

- that molecule means **POSITIVE** answer.

Thank you for your attention! :)