Jan. 2004, Vol.19, No.1, pp.98–inside back cover *J. Comput. Sci. & Technol.*

Trends in Computing with DNA

Nataša Jonoska

University of South Florida, Department of Mathematics, Tampa, FL 33620, U.S.A.

E-mail: jonoska@math.usf.edu

Received May 6, 2003; revised September 22, 2003.

Abstract As an emerging new research area, DNA computation, or more generally biomolecular computation, extends into other fields such as nanotechnology and material design, and is developing into a new sub-discipline of science and engineering. This paper provides a brief survey of some concepts and developments in this area. In particular several approaches are described for biomolecular solutions of the satisfiability problem (using bit strands, DNA tiles and graph self-assembly). Theoretical models such as the primer splicing systems as well as the recent model of forbidding and enforcing are also described. We review some experimental results of self-assembly of DNA nanostructures and nanomechanical devices as well as the design of an autonomous finite state machine.

1 Introduction

As we celebrate the 50th year since the discovery of the structure of DNA, extraordinary advances in genetics and biotechnology arrive on a daily basis. At the same time we are witnessing research developments that employ DNA in a completely new way treating this molecule of life as a nanomaterial for computation. The first conception of using DNA for computation ripened in the mid 80's with the first theoretical model of splicing systems introduced by Head[1]. These ideas came to full power with Adleman's seminal experiment[2] which solved a small instant of a combinatorial problem using solely DNA molecules and biomolecular laboratory techniques. The impact of these first ideas on many researchers can be observed by the numerous theoretical results and innovative experimental solutions that followed. Results of these studies are considered rather significant that much of the research has been published by leading scientific journals such as *Science* and *Nature*. The research has spurred new scientific interactions and opened connections between mathematics and computer science from one side, and molecular biology, nanotechnology and biotechnology from another.

The research in biomolecular computing has already taken many different pathways (theoretically and experimentally), such that, in attempt to describe some of these ideas, this paper will end up presenting just a small subset of the results. Even more, the choice of the content and the results that are covered are colored by the bias of the author. The reader is advised to consult the Proceedings of the annual meetings on DNA-based computers, currently in its ninth year, where most of the researchers present their results (see [3–10]).

In these highly interdisciplinary studies, theory and experiments are tightly interlaced and this paper contains a little bit of both flavors. It is assumed that the reader is familiar with basic biomolecular techniques, but for a reference, the first section gives a brief introduction to the notions used in the following sections. It is also assumed that the reader is familiar with mathematical writing as well as with the basic ideas of theoretical computer science which are used in describing the theoretical models.

The paper starts with a rather brief introduction to some notions from molecular biology and biotechnology. A short description of DNA molecules used in successful construction of nanostructures and nanomechanical devices is given in Subsection 2.2. The main ideas exploited with the first experimental success are described in Section 3. One of the ultimate problems for trying any new approach for DNA-based computation has been the satisfiability problem (SAT). Section 4 describes three different ideas for a solution of this problem with DNA: using linear duplex molecules, self-assembly of DNA tiles and graph self-assembly with junction molecules. There are numerous theoretical models for DNA-based computers and it is really difficult to describe even a small subset of them in a scope of this article. Hence, in Section 5 we concentrate on the very first model, the simple splicing system and one of the most recent ones, forbidding-enforcing systems. The section on nanodevices, Section 6, describes three very significant

This work has been partially supported by the National Science Foundation of USA under Grants No.EIA-0086015 and No.EIA-0074808.

experimental results. The first one is a DNA-based mechanical device that uses DNA "fuel" strands to change from one state to another. The second device is a DNA-based "switch" that uses the idea of fuel strands to switch between two positions. The section ends with a description of an autonomous finite state machine. It employs a restriction enzyme coupled with clever encoding that "reads" the input and recognizes certain strings. The paper ends with a few concluding remarks.

2 Biotechnology

Briefly we recall the basic DNA structure and the actions of several types of enzymes. A more thorough and not very technical description of the structure of DNA and the operations performed by enzymes can be found in [11, 12] (see also [13]). Detailed laboratory protocols can be found in [14].

Information in a DNA molecule is stored in a sequence of nucleotides, also called by their chemical group, base, A, G, C, T (*adenine, guanine, cytosine* and *thymine*) joined together by phosphodiester bonds. In the case of RNA the thymine is substituted with *uracil* U. A single strand of DNA, i.e., a chain of nucleotides, has also a "beginning" (usually denoted by $5'$) and an "end" (denoted by $3'$), and so, the molecule is oriented. A chain of nucleotides is called *oligonucleotide* or simply just *oligo*. By the well known Watson-Crick complementarity, A is complementary to T and C is complementary to G. A double stranded DNA is formed by establishing hydrogen bonds between the complementary bases of two single stranded molecules that have opposite orientation. This process is usually called *hybridization* or *annealing*.

2.1 Enzymes and DNA Operations

• *Polymerase* (used in operation "amplify" or "detect"). DNA polymerases are enzymes that synthesize DNA. With these enzymes, a DNA strand can be duplicated or extended. One of the commonly used protocols in molecular biology is the so-called *polymerase chain reaction* (PCR). This reaction detects certain DNA sequences and synthesizes a large number of such molecules from an existing pool of molecules. This method is used to detect (extract) a certain sequence within a large mix of molecules or to amplify.

• *Restriction enzymes* (*endonucleases*) (used in operation "separate" or "cut"). Such an enzyme recognizes a specific sequence of nucleotides in a double stranded DNA molecule and cuts the molecule in two pieces by destroying the phosphodiester bonds at specific places of the two strands. Different enzymes recognize different sequences of nucleotides, and even if they recognize the same sequence, they may cut the molecule in a different way.

• *Ligase* (used in operation "glue"). It is said that a double stranded DNA molecule has a *nick* if the phosphodiester bond between two consecutive nucleotides within one of the strands is broken. A ligase is an enzyme that closes the nicks, i.e., recovers the broken phosphodiester bonds in a double stranded DNA.

• *Length* (*weight*) *selection* (used in operation "separate" or "detect"). A technique called "gel electrophoresis" separates DNA molecules by their weight. The DNA is negatively charged and after being placed in a small well of a gel in an active electric field, it slowly moves toward the positive side. Larger (heavier) molecules move slower and smaller molecules move faster. The portion of the gel that contains molecules with the desired length can be cut out of the gel, DNA purified, and then used in subsequent experiments.

2.2 DNA Molecules Used in Nanostructures

In nature, DNA appears as linear double stranded molecule (in eukaryotes) but also it can be in a circular form (mostly in viruses and bacteria, prokaryotes). Circular DNA can be obtained also in a laboratory by joining (ligating) the ends of a linear DNA. Such molecules are used in several models of DNA-based computers (e.g., [12], Chapter 9). Circular molecules have been used as building blocks for DNA knots. In theory, virtually any knot can be constructed using right-handed B-DNA for negative crossings and left-handed Z-DNA for positive crossings[15–18]. Many catenas and linkages of DNA molecules are known, but just recently, the first Borromean DNA rings were assembled using B and Z DNA[19].

In DNA-based computing the complementarity of the nucleotides ($A \leftrightarrow T$, $G \leftrightarrow C$) is one of the basic properties used for encoding information as a tool for computation, as well as for obtaining DNA nanostructures. Besides the linear duplex DNA, there are two additional DNA building blocks frequently used, junction molecules and DNA prototiles made of DX or TX molecules.

Fig.1. A four armed branched junction molecule, and a DNA tile made of a triple cross over molecule.

Junction molecules are fairly well understood. These molecules were used in construction of DNA polyhedra, a quadrilateral, a truncated octahedron and a cube[17,20,21]. The k-armed branched molecules (k is a natural number $\geqslant 2$) seem to be suitable for graph construction. An example of 4-armed branched molecule is presented in Fig.1 to the left. In this figure, the double helix of the molecule is not presented. Hydrogen bonds between the anti-parallel, complementary Watson-Crick bonds are depicted as dotted segments between the strands. Polarity of the DNA strands is indicated with arrowheads being placed at the 3′-end. The angles between the "arms" are known to be flexible. If we allow each "arm" to be over 200 or 300 base pairs long, then the "arms" of this molecule become rather flexible and we deliberately show them curved. On the other hand, short arms of two or three helical turns (one helical turn is about 10.5 bases) would provide quite rigid structure. Such rigid structures show high potential for constructing three dimensional crystals. Two dimensional arrays made of rigid structures have been reported in [22]. The 3′-ends can be extended such that each arm ends with a single stranded portion. This single stranded part, also called "sticky end", can anneal to its Watson-Crick complement once placed in a test tube. Construction and properties of these molecules are fairly well understood[23,24] such that their potential for utilizing them in a more complex structures is becoming rather feasible. Recently, general non-regular graphs have been successfully constructed by using junction molecules for the vertices and duplex molecules for the edges[25].

DNA prototiles. Another very important step toward constructing three dimensional DNA crystals was made by the design and the assembly of a two dimensional array made of DNA tiles[26]. The construction of these arrays was enabled by the use of double (DX) and triple (TX) cross over molecules that act as tiles. These molecules are double or triple duplex molecules (two or three double helices) such that DNA strands interchange between different helices. An example of a triple cross over molecule is presented in Fig.1 to the right (figure obtained from [27]). The 3′-ends are indicated with an arrow and they may be extended to be used as sticky ends such that connecting TX molecules in a two-dimensional array is possible[28]. These DNA tiles made of DX and TX molecules have been initially designed in Seeman's laboratory at New York University and now are used by several groups in Caltech, Duke and University of Southern California.

3 Beginning

In his experiment, Adleman[2] solved a small instant of a combinatorial problem known as Hamiltonian Path Problem for a directed graph. The Hamiltonian Path Problem (HPP) asks whether for a given (directed) graph G there is a path from one vertex (denote it with v_{in}) to another vertex (denote it with v_{out}) that visits every vertex exactly once. If such path exists then it is called Hamiltonian.

In the Adleman's experiment, the edges of G are represented by single-stranded DNA oligos made of twenty (randomly chosen) nucleotides. The vertices in the graph are also oligos of 20 nucleotides having the first 10 nucleotides complementary to the last 10 of the incoming edge, and the other 10 being complementary to the first 10 nucleotides of the outgoing edge (see Fig.2). A path $e_1 \cdots e_k$ of length k in G is represented by a double-stranded DNA molecule of length $20k$ base pairs with ten nucleotides (single-stranded) overhang from each end. So, if a Hamiltonian path exists, then it has to be represented by a double stranded molecule of length $20n$ where n is the number of vertices. The experiment that solved the problem had the following key elements:

1) encode the information about the graph into DNA strands;

2) use self-assembly of the molecules led by the Watson-Crick complementarity to generate a large library of strands that encode paths in the graph;

3) use known biomolecular techniques (ligation, gel electrophoresis, PCR, affinity separation) to extract the right solution, i.e., to extract the molecules that represent paths from v_{in} to v_{out} and visit every vertex exactly once;

4) provide a way (gel-electrophoresis) to read out the output.

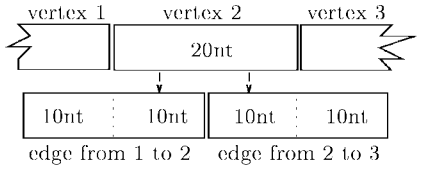

Fig.2

The problem that was chosen for this experiment was a well known NP-complete problem that is generally "intractable" in the sense that for a relatively modest size of a graph, with any known algorithm, an impractical computer time is needed for solution. The Adleman's approach to this problem is not much different then a brute force search, but the way it was encoded and solved, the use of *massive parallelism*, *self-assembly* and the use of *non-determinism* made it very novel, inspiring and a base for innovative ideas both experimentally and theoretically.

Several theoretical models based on the lab protocols used in the Adleman's experiment can be found in the literature (e.g., [12, 29, 30]). They all use more or less the same set of operations: *merge, separate, detect, amplify* etc. and they all can feasibly be executed by a robotic system. In [30], Lipton showed that, using these operations, the satisfiability problem (see Section 4) for propositional formulas can be solved and consequently a large set of problems can be solved by DNA. In [31], it is shown how these operations can be used to break the Data Encryption Standard (DES). Approximately one gram of DNA is needed and using robotic arms (assuming each operation to last one minute) breaking DES is estimated to take five days. The most significant in the analysis for breaking DES is that the success is quite likely even with, at this point unavoidable, large number of errors within the lab protocols.

The big drawback in the Adleman's and Lipton's approach is the need of a very large pool of initial molecules that have to be generated in order to assure correct solution to the problem. For a larger graph, say a modest size of 200 vertices, one needs "DNA more than the weight of the Earth" (see [32]). The subsequent studies have concentrated on developing algorithms such that not necessarily all of the potential solutions are constructed at once (e.g., [33, 34]). However, scaling up the proposed models to a larger and practically significant problems remains one of the most difficult problems in using DNA for computation.

4 Solving SAT

The SAT problem is one of the standard examples of NP-complete problems, and in fact, many researchers have suggested various ways to solve this problem by different biomolecular techniques. Here we show three of these approaches, using linear duplex molecules, using DNA tiles and using DNA graph structures. Although it is now clear that solution of large scale combinatorial search problems, most probably, will not be the future of DNA-based computations, new methods, models and techniques are described very often through proposals for a solution to this problem. The method that uses linear duplex molecules is the first one that was proposed by Lipton[30]. It requires linear number of laboratory steps (regarding the size of the formula), and the experimental results are most successful.

We start with a definition of the problem. Let $A = \{a_1, a_2, \ldots\}$ be a set of Boolean variables. A *clause* is a formula $C = b_1 + b_2 + \cdots + b_k$ where $+$ is the logical "or" and each b_i is a variable or a complement to a variable in A.

A *conjunctive logical formula* is a formula of the form: $\alpha = C_1 \cdot C_2 \cdot \ldots \cdot C_r$ where each C_i is a clause.

The *satisfiability problem* (SAT) asks whether for a given logical formula, α, there is an assignment of $\{True, False\}$, i.e., $\{1, 0\}$ to the variables in α that would assign α the value of T, i.e., 1.

Consider the example:

$$\alpha = (\bar{x} + y + \bar{z})(x + \bar{y} + z)(\bar{x} + \bar{y} + z). \qquad (*)$$

This formula $(*)$ has value 1 for the assignments $(x, y, z) \in \{(0,0,0), (1,1,1), (1,0,0), (0,1,1), (0,0,1)\}$ and 0 for any other assignment. We will follow this example for the discussion below.

4.1 Solving SAT with Linear Duplex Molecules

Right after the initial Adleman's report of a successful experimental solution of an instance of HPP, Lipton[30] realized that a large class of computationally hard problems can be solved using es-

sentially the same techniques. (1) Encode the variables and their truth values within DNA strands. (2) Use Watson-Crick complementarity and recombinant DNA operations to separate the solution. This method relies on availability of a large pool of bit strands encoding the variables and their truth values. Using this approach now there are quite few experimental solutions of SAT.

Landweber and her group in Princeton used RNA to encode the bit strands[35]. They solved a 3×3 knight problem which is equivalent to a solution of a SAT problem with nine variables. Currently this group is developing an automated system using a microfluidic device to separate the strands. Such a device was initially reported by the group of McCaskill[36−38], at the GMD-National Research Center for Information Technology in Germany and currently it is being developed by Van Noort and Landweber in Princeton, USA.

The research group at University of Southern California utilized hybridization of short sticker single stranded DNA strands to encode the Boolean variables similarly as was proposed by Lipton. The solution is obtained by separating strands that encode assignments of the variables that satisfy the given formula. This separation employs complementary stickers and requires sensitive and error-resistant techniques that separate a small set of specified molecules from a large pool of distinct molecules. An automated mix of thermo-cycler and gel-eloctrophoresis technique was developed for this purpose and in a major breakthrough they achieved to solve an instance of SAT with 20 variables[39].

At the same time the Japanese group led by Suyama reported that they are also capable of solving an instance of SAT with 20 variables. Their algorithm is based on the idea that not all possibilities for the truth values of the variables are encoded at once, but the solution is slowly built up according to the problem. This approach was first reported by Suyama in [33] and recently the robots designed for implementation of these techniques were presented in [40].

Hagiya *et al.*[41] presented another algorithm for solving SAT, the so-called SAT engine which uses a hairpin formation of a single stranded molecule as a test for inconsistent value assignment of the variables. Each strand contains values of variables that are contained in one clause, such that opposite values of the variables are encoded with complementary nucleotides. When such inconsistent assignments are made, the DNA strand forms a hairpin and is cut by a restriction enzyme employing the recognition site inserted in the value sequence. Hagiya also initiated the model of *Whiplash PCR* that uses a single molecule and its hairpin formation to simulate a finite state machine[42]. The method was subsequently improved by Winfree[43].

4.2 Solving SAT with DNA Tiles

The methods mentioned in the previous section have been shown to be relatively successful experimentally, but they all require increase in the number of laboratory operations (linear with the size of the formula). Although the recombinant DNA operations are being automated and robotized, they still are not precise. In each laboratory protocol one assumes less than 100% yield, often less than 80%. As yield decreases with each additional protocol, it becomes valuable to have models and computational techniques that require little outside mediation or human interaction. Scaling up of the algorithms comes as another problem, in particular, generating a combinatorial library that contains all possible solutions (as was suggested by Lipton, and done by Landweber and Adleman) becomes quite impractical even for a modest size problem. Many authors have suggested use of the self-assembly of DNA tiles (DX or TX molecules) as a possible model of computation. This model uses one-pot reactions, and is based solely on the Watson-Crick complementarity of the sticky ends.

First to realize that the self-assembly of DNA tiles is capable of simulating a Universal Turing machine was Winfree[44]. He showed that the four sticky ends in a DX molecule can be considered as four sides of a Wang tile[45]. By tiling the plane with Wang tiles one can simulate the dynamics of one-dimensional cellular automata and with this, a Universal Turing machine. Two dimensional arrays made of DX and TX molecules have been reported by the Seeman's laboratory[28,44]. There are no successful experimental self-assembly of two dimensional programmable arrays, but the laboratory of Winfree at Caltech reported good progress toward assembly which simulates dynamics of a cellular automaton producing the Sierpinsky triangle[46].

Besides two dimensional arrays, a linear array of DNA tiles can also be used as a computational model. In [47] the authors show solutions of various different computational problems using string assembly of DNA tiles. They prove that, theoretically, the languages that are generated by linear assembly of finite DNA tiles correspond to the ET0L systems. Fig.3 shows the idea of using linear (string) DNA tile assembly to solve SAT. The

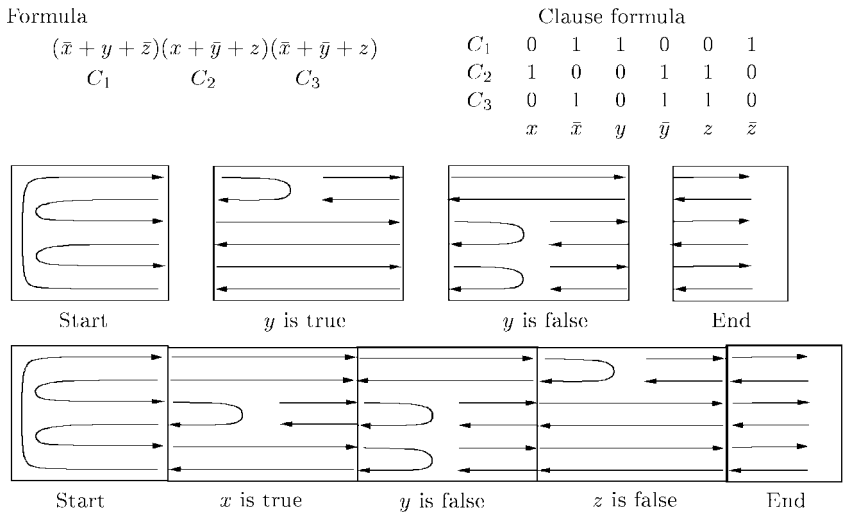

Fig.3. Solving a 3-variable SAT problem by linear assembly of DNA tiles. The double helical structure of the strands is not presented for simplicity. The string tile assembly at the bottom row represents one possible solution.

width of each tile (i.e., the number of DNA double helices) is equal to the number of clauses in the formula. There are two tiles for each variable a_i: a "true" a_i and a "false" $\bar{a}_i$ tiles. For each clause C_j in which a_i appears, "true" a_i tile has a hairpin structure at that position. Similarly with the appearance of $\bar{a}_i$ and the "false" $\bar{a}_i$. For example, in Fig.3, the formula α has three clauses and the tiles have three helices, i.e., they are TX molecules. The variable y appears in clause C_1 as y and in C_2 and C_3 as $\bar{y}$. Hence the "true" y tile has the first helix as a hairpin and the "false" $\bar{y}$ tile has hairpins in the second and the third helices. The hairpins "cap off" the clauses which are satisfied by the truth value of the variable of the tile. In this way, the formula α is satisfied if and only if a circular molecule is constructed during the assembly.

This idea of having one single strand that represents solution to the given problem was employed by Mao *et al.*[48] where a linear self assembly of TX molecules that perform binary addition (cumulative exclusive OR) of a sequence of bits was obtained experimentally. The result could be read out within a single strand. This computational method seems to be promising since the "programming" of the tiles is rather straightforward, the computation is performed by self-assembly without outside mediation and the steps are executed in a massive parallel way. This reduces the human interaction to a minimum, but unfortunately it does not solve the scaling problem.

4.3 Solving SAT by Graph Self-Assembly

DNA is a three dimensional molecule and the Watson-Crick complementarity can be used to assemble three dimensional structures. Using junction molecules, one can imagine assembling arbitrary graphs or other three-dimensional structures. The first successes along these lines are already one decade old. Chen and Seeman used junction molecules to construct a cube and a truncated octahedron[20,21]. In [49, 50], the authors show that many combinatorial problems can be solved by graph assembly. The idea is to encode the problem in duplex molecules and branched junction molecules such that the graph can self assemble if and only if there is a solution to the problem. We illustrate this technique here with the solution of SAT as presented in [25].

Observe that for each formula α with variables $a_1, \ldots, a_k$ and clauses $C_1, \ldots, C_r$ we can associate a graph $G(\alpha)$ defined in the following way. The vertices of G are $\{a_1, \ldots, a_k, \bar{a}_1, \ldots, \bar{a}_k, C_1, \ldots, C_s\}$. The set of edges contains $\{a_i, \bar{a}_i\}$ for $i = 1, \ldots, k$. If a clause C_j contains atoms b_{j1}, b_{j2}, b_{j3} where b_{ji} is either a variable or a negation of a variable, then $G(\alpha)$ contains the edges $\{C_j, b_{j1}\}, \{C_j, b_{j2}\}, \{C_j, b_{j3}\}$. The graph $G(\alpha)$, corresponding to the formula $(*)$, is depicted in Fig.4.

The idea for solving 3-SAT is to encode the problem into junction DNA molecules such that the

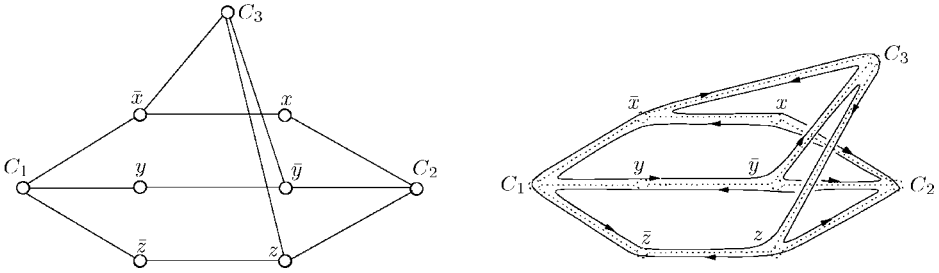

Fig.4. The graph $G(\alpha)$ corresponding to the formula $(*)$ is presented on the left. To the right is a possible DNA graph structure representing the graph $G(\alpha)$.

graph $G(\alpha)$ is self-assembled according to its Watson-Crick coding if and only if the formula α has a solution. Each clause C (containing atoms, say x, y, z) is represented with a 3-junction molecule such that the arms of the molecule end with a single stranded extension containing three parts of the encoding. The first and the third parts of the encoding are atom specific, and the middle part is value (True $= 1$, or False $= 0$) specific.

Since a clause has value "true" if at least one of the atoms in the clause has value true, seven different molecules for each clause are needed (each molecule encodes one of the seven possibilities of the atom values that assign value true to that clause).

Each variable and its complement in the graph $G(\alpha)$ is represented with two adjacent vertices. The degree of these vertices equals the number of clauses each of the variables or their complement belongs to. In the DNA graph, this is represented with two junction molecules, call them J_1 and J_2 glued together. One of these junctions, say J_1, represents the variable, say x, and the other J_2, represents its negation $\bar{x}$. As with the clause building blocks, the single stranded encodings have three parts. The truth value encoded in the arms of J_1 is opposite to the truth value encoded in the arms of J_2. For the formula $(*)$ and the variable x, one building block is depicted in Fig.5.

To form a DNA graph corresponding to $G(\alpha)$, all clause molecules and all variable building blocks are combined and their compatible ends are allowed to form double-stranded DNA. Once formed, the molecules are locked together by sealing all open "nicks" with DNA ligase.

A possible DNA structure that corresponds to the graph $G(\alpha)$ for the formula α in Example $(*)$ is depicted in Fig.4 to the right. Once ligated, it is one circular single stranded molecule, i.e., a knot. Such knot structures have been used to study recombination enzymes[51], and similar techniques for detecting the knot structures might be useful here. There are graphs, however, that do not form a knot, but rather form links (multicomponent circles), and partially formed graphs may contain smaller knots and links as well. This question was addressed in [52].

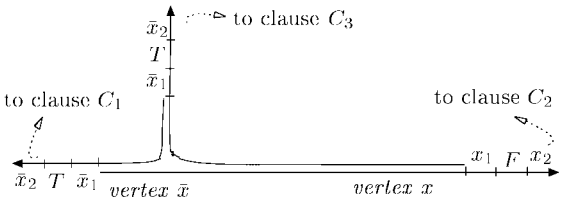

Fig.5. A building block for variable x and $\bar{x}$ in $(*)$. Since $\bar{x}$ appears in two clauses, it is encoded with a 3-junction. The variable x appears in only one clause, so it is represented with a duplex molecule, one side of which is connected to one arm of the 3-junction representing $\bar{x}$.

5 Theoretical Models

As computer scientists and mathematicians became aware of the laboratory possibilities involving DNA molecules, the theoretical models for DNA-based computations started to develop and they appeared in many different flavors. The very first one that appeared few years before the Adleman's experimental result, was introduced by Head[1]. This model initiated the study of H-systems (i.e., splicing systems) and the early research following this initial paper was concentrated on characterizing H-systems and proving that they generate regular languages. Head, in [53], suggested use of circular (cyclic) strands within splicing systems and soon after, the interest for H-systems as a theoretical model for DNA-based computation exploded. It took about ten years for the researchers to understand the computational power of these systems

[54]. Many variations of H-systems from the original definition have been considered in the literature and for more detailed description the reader is advised to consult with [11, 12]. In this paper, we describe the main idea behind splicing systems by considering "simple" splicing.

Another rather new theoretical model motivated by chemical reactions among different sets of molecules are the so-called "forbidding-enforcing" (fe-) systems introduced in [56]. Being rather new, there are many things that need to be yet understood for this model. At this point we do have a way to do computation (through the so-called Γ-trees described in [56]) and we have some understanding about the topological properties of these systems[57,58]. Most recently, fe-systems have been considered within the concept of membrane computing[59]. In Subsection 5.2 we present the definition for this model and an example for the solution of 3-SAT based on forbidding and enforcing.

There are many other theoretical models for DNA-based computations, most of them having computational power equivalent to the Universal Turing machine. These include sticker systems[12,60], insertion-deletion systems[61], Watson-Crick automata[12,62] etc. but will not be included in the scope of this presentation.

5.1 Splicing Systems

In order to illustrate the splicing idea, we concentrate on the simplest example, the "simple" splicing system. This example might not be very interesting from computational power point of view (the language generated by a simple splicing system is strictly locally testable), but the splicing systems started with this model and its relation to the action of the endonucleases on DNA can be explained in a vivid way. In particular, just recently, an algebraic characterization of this type of splicing system was obtained, the first one for any class of splicing systems[63].

Restriction enzymes (endonucleases) of type II recognize specific sites (recognition sites) of a double-stranded DNA molecule and cut the molecule (by destroying the phosphodiester bonds), often in a way that leave a small single-stranded overhang.

Once cut, molecules with complementary overhangs can join together and the phosphodiester bonds can be re-established with an aid of a ligase. The question rises, "what words (strings of DNA molecules) can be obtained in a single tube by allowing a set of restriction enzymes and a ligase to react?". This question was first considered by Head[1] and then followed by extensive studies of this and extensions to this H-system by other researchers such as R. Freund, Gh. Paun, G. Rozenberg and others[11,12]. The theoretical set-up is as follows.

Let $\mathcal{A}$ be an alphabet, $\mathcal{M} \subseteq \mathcal{A}$, and $L \subseteq \mathcal{A}^*$, and let $\sigma_{\mathcal{M}}(L)$ be defined by $z \in \sigma_{\mathcal{M}}(L)$ if and only if there exist $x, y \in L$ such that for some $c \in \mathcal{M}$, $x = x'cx''$, $y = y'cy''$, and $z = x'cy''$. The set of *factors* or *subwords* of a language L is denoted with $\underline{\mathrm{Sub}}(L)$ i.e., $\underline{\mathrm{Sub}}(L) = \{x : \exists y \in L, y = z_1 x z_2 \text{ for some } z_1, z_2 \in A^*\}$. A *simple H-system* is a triple $H = (\mathcal{A}, S, \mathcal{M})$, where $\mathcal{A}$ is a finite alphabet, $S \subseteq \mathcal{A}^*$ is a finite set of initial words, called *axioms*, and $\mathcal{M} \subseteq \mathcal{A}$. The language generated by H is $L(H) = \bigcup_{i=0}^{\infty} L^{(i)}$, where the sets $L^{(i)}$ are defined recursively by $L^{(0)} = S$ and $L^{(i+1)} = L^{(i)} \cup \sigma_{\mathcal{M}}(L^{(i)})$, $i \geqslant 0$.

In order to see that simple H-systems generate strictly locally testable languages we follow [64], and recall the definition of constants for a language. A word c is a *constant* for a language L if and only if $xcy' \in L$ whenever $xcy, x'cy' \in L$. For a given simple H-system, $H = (A, S, \mathcal{M})$, it is clear that if $c \in \mathcal{M}$, then c is a constant for $L(H)$. But even more is true: if $u \in \underline{\mathrm{Sub}}(L(H))$ is such that $\underline{\mathrm{Sub}}(u) \cap \mathcal{M} \neq \emptyset$, then u is a constant for $L(H)$. For let $u = u_1 c u_2$, where $c \in \mathcal{M}$, and suppose that $xuy, x'uy' \in L(H)$. Since $c \in \mathcal{M}$, the word xu_1cu_2y' is in $\sigma_{\mathcal{M}}(L(H)) = L(H)$, (consider xu_1cu_2y and $x'u_1cu_2y'$), so $xuy' \in L(H)$. It follows that if $k = \max\{|u| : u \in \underline{\mathrm{Sub}}(S), \underline{\mathrm{Sub}}(u) \cap \mathcal{M} = \emptyset\}$, then every word of length greater than k is a constant for $L(H)$. This is the well known characterization of strictly locally testable languages[65]. (See also Theorem 4.6 in [11] and Subsection 7.5 in [12].)

We now return to the action of endonucleases on DNA segments. Assume that we have a finite number of distinct double-stranded DNA segments, and an infinite supply of each segment. Assume further that we have a finite number of restriction endonucleases and a ligase. Denote by E the set of restriction endonucleases and consider the alphabet $\mathcal{A} = \{A, C, G, T\}$. Write the sequences of double-stranded DNA segments in direction $5'$ to $3'$, and consider them as elements of $\mathcal{A}^*$. For example, if we have a segment $\begin{smallmatrix} {}^{5'}AAGCCT^{3'} \\ {}_{3'}TTCGGA_{5'} \end{smallmatrix}$, then we include $AAGCCT$ and $AGGCTT$. For each restriction endonuclease ϵ, consider its restric-

tion site ρ_ϵ (again in direction $5'$ to $3'$) and denote the overhang produced by the action of the enzyme with α_ϵ. For example, for *Eco*RI, the restriction site is $\rho_{Eco\mathrm{RI}} = GAATTC$ and the overhang is $\alpha_{Eco\mathrm{RI}} = AATT$. Denote with S' the set of all initial DNA segments, written as words in $\mathcal{A}^*$, as described above. Let $\{\alpha_1, \ldots, \alpha_k\}$ be the set of all distinct overhangs produced by the restriction enzymes in E. For $i = 1, 2, \ldots, k$, let $\bar{\alpha}_i$ be a new symbol encoding the sequence α_i, and let $\Sigma = \{\bar{\alpha}_1, \ldots, \bar{\alpha}_k\}$. Extend $\mathcal{A}$ to $\bar{\mathcal{A}} = \mathcal{A} \cup \Sigma$ and define a simple H-system $\bar{H} = (\bar{\mathcal{A}}, S, \Sigma)$ where

$$S = \{xu\bar{\alpha}_i u'y \mid xu\alpha_i u'y \in S' \text{ and there exists } \epsilon \text{ such that } \rho_\epsilon = u\alpha_i u' \text{ and } \alpha_\epsilon = \alpha_i\}.$$

It is easy to see that the language $L(\bar{H})$ generated by the simple H-system $\bar{H}$ equals the set of distinct molecules produced by the action of a ligase and the set of endonucleases on the set of molecules S'.

5.2 Forbidding and Enforcing

Models for DNA computation based on classical concepts in formal language theory employ determinism in various forms, such as rewriting techniques and grammar systems, and for many can be proven that they have universal computational power. However, the deterministic way of defining languages by "everything that is not allowed is forbidden" employed by these models does not necessarily correspond to the various non-deterministic ways that biomolecules act within one biochemical reaction. Motivated by this non-determinism (considering molecules as strings, and sets of molecules as languages) the authors in [56, 57] use boundary conditions of *forbidding and enforcing* to introduce another way of defining classes of languages. This new concept is best described with the phrase "everything that is not forbidden is allowed". In this sense, the forbidding boundary conditions imply that certain words are not allowed in the language, but any language that does not contain words that are forbidden is allowed. This corresponds to a biochemical condition when certain molecules cannot "survive" in a given biochemical environment. Similarly, the enforcing condition says that with presence of a specific set of words, some other words must be present as well, and hence, any language that contains these words is allowed in the family. Again, biochemically, the enforcing conditions provide a model where in certain conditions presence of some molecules imply presence of other (new) molecules. As in any biochemical reaction, the resulting set of molecules (i.e., resulting language) can belong to a family of possible sets, all satisfying the initial boundary conditions. It is shown in [56, 57] that many computational problems such as SAT and Hamiltonian Path Problem can be solved using fe-systems and we describe a possible solution of SAT here.

A pair of finite sets (X, Y) $(X, Y \subset A^*)$ is called an *enforcer*, and we say that a language L satisfies the enforcer (X, Y) if $X \subset L$ implies that $Y \cap L \neq \emptyset$. In that case we write $L \underline{sat} (X, Y)$. Note that if $X \not\subset L$, then (X, Y) is trivially satisfied. When $X = \emptyset$, the enforcers are called *brute* enforcers. If for an enforcer (X, Y) we have that $X \cap Y \neq \emptyset$, then every language L satisfies (X, Y). Such enforcers are called *trivial*. If E is a set of enforcers, then $L \underline{sat} E$ if it satisfies every enforcer in E. The family of languages defined by the enforcing set E is denoted with $\mathcal{L}(E)$ and consists of all languages that satisfy the enforcer E.

A finite set of words $F \subset A^*$ is called a *forbidder* and a language L is *consistent* with a forbidder F if $F \not\subset \underline{\mathrm{Sub}}(L)$ where $\underline{\mathrm{Sub}}(L)$ denotes the set of all subwords of words in L. In this case we write $L \underline{con} F$. If $\mathcal{F}$ is a set of forbidders, then we write $L \underline{con} \mathcal{F}$ if L is consistent with every forbidder in $\mathcal{F}$. The set of all languages consistent with a forbidding set $\mathcal{F}$ is denoted with $\mathcal{L}(\mathcal{F})$. An *fe-system* is a pair $(\mathcal{F}, E)$ defining a class of languages $\mathcal{L}(\mathcal{F}, E) = \{L \subseteq A^* \mid L \underline{sat} E \text{ and } L \underline{con} \mathcal{F}\}$ and so $\mathcal{L}(\mathcal{F}, E) = \mathcal{L}(F) \cap \mathcal{L}(E)$.

5.2.1 *Solving SAT with fe-Systems*

Consider the formula α presented with $(*)$ in Section 4. The clauses of the formula are $C_1 = \bar{x} + y + \bar{z}$, $C_2 = x + \bar{y} + z$ and $C_3 = \bar{x} + \bar{y} + z$. Define an fe-system as $\Gamma = (\mathcal{F}_1 \cup \mathcal{F}_2, E)$ such that

$$E = \{(\emptyset, \{1x, 0x\}), (\emptyset, \{1y, 0y\}), (\emptyset, \{1z, 0z\})\}$$
ensure all possible truth values
$$\mathcal{F}_1 = \{\{1x, 0x\}, \{1y, 0y\}, \{1z, 0z\}\}$$
define unique value for each variable
$$\mathcal{F}_2 = \{\{1x, 0y, 1z\}, \{0x, 1y, 0z\}, \{1x, 1y, 0z\}\}$$
ensure that each clause is true

Now every language that belongs to $\mathcal{L}(\Gamma)$ has to satisfy the set of enforcers E, i.e., each variable has to be assigned at least one truth value. The forbidding set $\mathcal{F}_1$ says that the subwords in each $L \in \mathcal{L}(\Gamma)$ cannot contain both truth assignments for a variable. The second forbidding set ensures

that the assignments that make each of the clauses "false" cannot be present in the language all at once. Since $\mathcal{L}(\Gamma)$ contains all languages that satisfy the enforcing and forbidding sets, we have that α is satisfiable if and only if $\mathcal{L}(\Gamma) \neq \emptyset$.

6 Nanomechanical Devices and Automata

The relatively predictable results by hybridization of two complementary DNA strands is one of the appealing reasons for using nucleic acids in nanotechnology. It has a minuscule size (about 2 nanometers in diameter), the single stranded parts can be considered as "sticky parts" that anneal to its complement with a formation of a double helix as a final result. The sticky parts provide predictable intermolecular interaction and the double helical structure provides a predictable final geometrical structure. Moreover, the nature has provided unique tools in the form of enzymes that allow us to have a tractable and controllable system. These properties are rather attractive for use in nanotechnology and for construction of nanodevices. We mention three recent developments in using DNA as a tool for construction of nanomechanical devices or as a model for a molecular finite state automaton.

6.1 DNA Actuator

Yurke[66] realized that when in a solution that contains DNA strands that are partially hybridized a strand that is completely complementary to one of the strands is introduced, then the hybridization of the complementary strands overcomes the partial hybridization (see Fig.6). This new complementary strand can be used as a "fuel" to "move" strands from one hybridization to another and with this to change the geometry of the the self-assembled structures in the tube.

The actuator like nanodevice uses DNA strands as a fuel and operates with the same principles. Two strands are assembled, strand A and strand B. The second strand B has a sequence length approximately double the length of the strand A (see Fig.7(a)). The strand A has both $3'$ and $5'$-ends complementary to the corresponding $5'$ and $3'$-ends of the strand B except few bases in the center part. The double stranded regions are less than 100bp and as such are rather stiff whereas the single stranded region is much more flexible. In this initial assembly the double stranded stiff regions are bent due to the central nucleotides that are free.

Motion on the actuator like device is induced by employing DNA fuel strands F that are complementary to the single stranded portion of the strand B. This makes a rigid type of DX molecule made of two duplex molecules. In this case the device is straighten (Fig.7(b)). The fuel strand is a bit longer than the single stranded portion of the strand B and it has non-hybridized single stranded part. The original (relaxed) state of the device is obtained by introduction of a complementary strand $\bar{F}$ to the strand F. The free single stranded portion of F anneals to $\bar{F}$ and then $\bar{F}$ starts to compete with the complex AB for binding with F (Fig.7(c)). Since $\bar{F}$ is a full complement of F and is firmly attached to its free single stranded portion, going through the process of three-strand branch

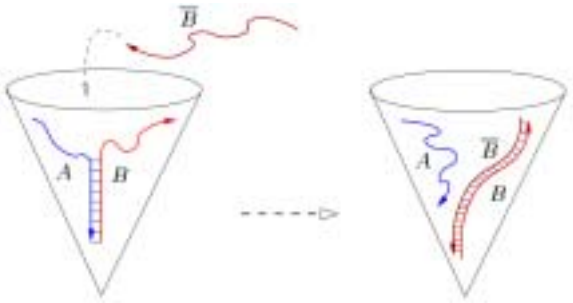

Fig.6. Assume a tube contains partially hybridized molecules A and B such that B has a portion that is single stranded and not annealed (figure to the left). If a complete complement to B, strand $\overline{B}$ is introduced, then the hybridization between B and $\overline{B}$ overcomes and strand A is released.

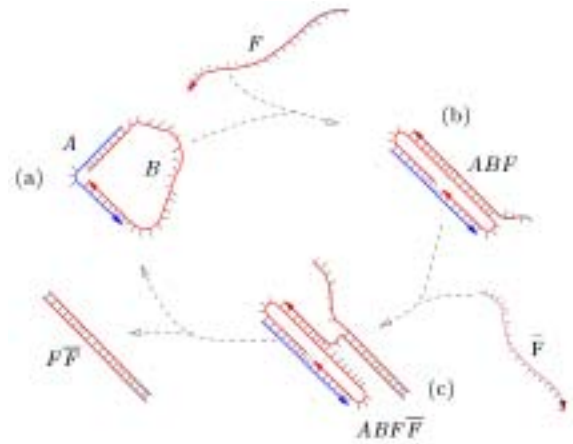

Fig.7

migration, $\bar{F}$ wins over the complex AB. It produces a waste product $F\bar{F}$ and relaxes the complex AB.

Yurke and his collaborators have developed several similar DNA-fueled nanomechanical devices[67,68]. One can envision such devices incorporated into two dimensional DNA arrays such that the whole surface could fold and open up. This combination has a great potential in developing nanomaterials and nanotemplates for circuit developments.

6.2 Two-State Switch

Using the same idea of fuel DNA strands, Seeman's laboratory developed a two-state switch[69]. This device is a combination of PX and JX_2 molecule that flip-flops between these two states as different fuel strands are added or removed (see Fig.8, this figure is adapted from [70]). One can consider a PX molecule as a double helix made of two DNA duplex molecules. This robust device, whose machine cycle is shown in Fig.8(a), is directed by the addition of set strands to the solution that forms its environment.

The set strands, drawn in green and yellow, establish which of the two states the device will assume. They differ by rotation of a half-turn in the bottom parts of their structures. The sequence-driven nature of the device means that many different devices can be constructed, each of which is individually addressable; this is done by changing the sequences of the red and blue strands where the green or yellow strands pair with them. As was the case with the fuel strands of the actuator, the green and yellow strands have short, single stranded extensions. The state of the device is changed by first binding full complements of green or yellow strands (fuel strands) and then removing them from solution (strands are biotin-tailed and can be removed by magnetic streptavidin beads). Adding the other strands changes the state of the device. Fig.8(b) shows that the device can change the orientation of large DNA trapezoids, as revealed by atomic force microscopy. The PX (green strand) state leads to parallel trapezoids and the JX_2 (yellow strand) state leads to a zig-zag pattern. This device has two very significant advantages. First, it is robust, and second, it is sequence dependent. One can envision several such devices, each addressable by different sets of strands used in a single nanostructure.

6.3 Finite State Automaton

One of the recent breakthrough in the ideas of using DNA for computation is obtained by another collaboration between a computer scientist E. Shapiro and a biochemist E. Keinan[71]. They treated a type II restriction endonuclease as a tool to change states in a finite state machine, such that together with a rather clever encoding of the states and the symbols, a successful model for recognizing formal languages was obtained. They used three main ideas: (a) use a restriction endonuclease *Fok* I; (b) encode a pair (state, symbol) with both, the sequence and the length of the segment; and (c) use accepting sequence for final readout.

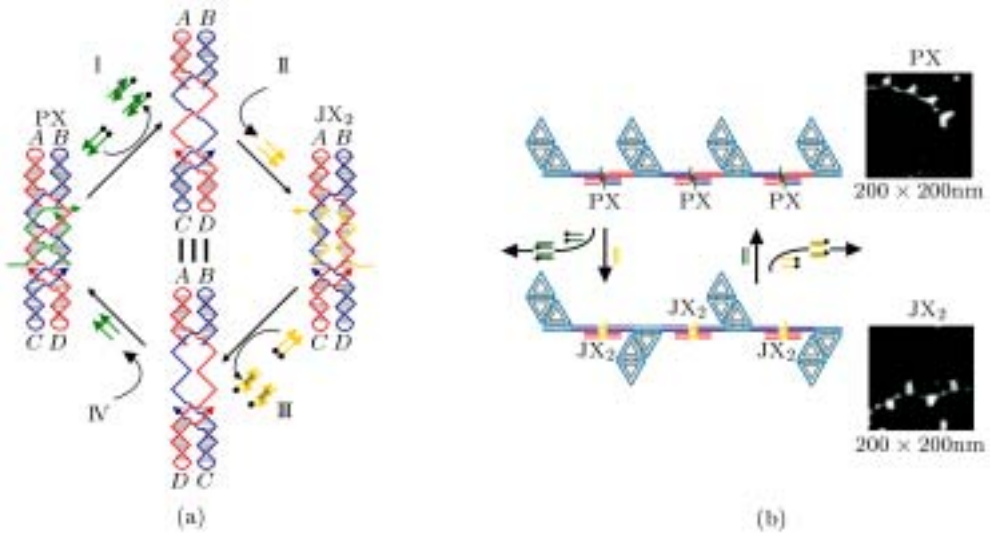

Fig.8

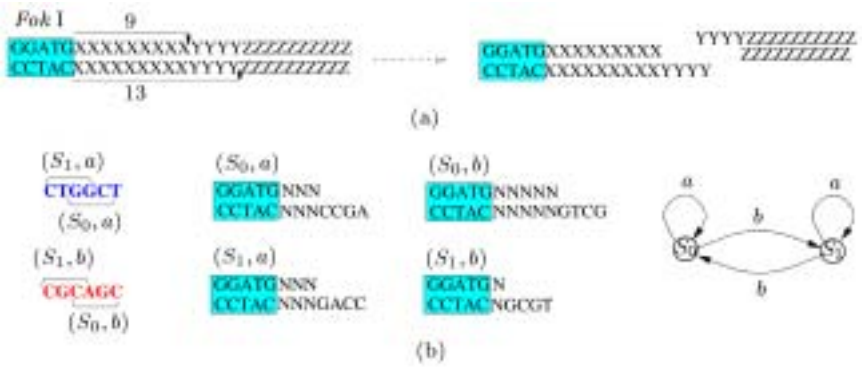

Fig.9. (a) Action of the enzyme. (b) State-symbol encodings, and transition molecules.

The enzyme that was chosen *Fok*I is such that it has a recognition site GGATG, but it cuts a double stranded molecule 9 and 13 bases away from the recognition site leaving a $5'$ overhang (see Fig.9(a)). The sequence of bases between the recognition site and the cutting position is completely irrelevant for the action of the enzyme. This is exactly the place where encoding of the symbols and the states of the automaton could take the whole advantage.

The authors demonstrated simulations for several automata with two states. One of these automata consists of two states s_0 and s_1 with the following transitions: $(s_0, a) \mapsto s_0$, $(s_1, a) \mapsto s_1$, $(s_0, b) \mapsto s_1$ and $(s_1, b) \mapsto s_0$, with s_0 being the initial state and the terminal state. This automaton recognizes (accepts) all words that have even number of b's. The (two) symbols are encoded with sequences of length 6 such that the first four of the sequence encode the state s_1 (i.e., in state s_1 reading the symbol) and the last four encode the state s_0 (Fig.9(b)). The state transitions are encoded with four short transition molecules, each starts with the recognition site of *Fok*I, followed with a sequence of computationally irrelevant base pairs. These transition molecules end with a $5'$ overhang of four bases complementary to the encoding the pair (*state, symbol*). The transitions for (s_0, a) and (s_1, b) have 3 base pairs which allow cuts of the input molecule to be at the same position of the 6 base pairs encoding an input symbol. The encoding of (s_0, b) has 5 irrelevant base pairs, and this "moves" the cut of the enzyme to the left leaving $5'$ overhang with the first 4 nucleotides of the 6 encoding an input symbol. This moves the reading of the next input sequence to encodings of s_1. The encoding of (s_1, b) has only one irrelevant base pair which "moves" the cut of the enzyme to the last 4 nucleotides (as are needed for state s_0). The input to the automaton is a strand that contains a restriction site for *Fok*I, seven (irrelevant) base pairs, a sequence of base pairs encoding a word with symbols a and b. At the end there is a terminal sequence which can anneal to the transition molecule that encodes a terminal state. The "automaton" changes its states and reads the input without outside mediation, and solely by the use of the enzyme. Fig.10 shows several transitions (computational steps) of the finite state machine. The initial experiment reported in [71] employed a ligase to 'glue' the transition molecules to the remaining of the input, but their subsequent study showed that this is not necessary[72].

This autonomous computational device is one of the most significant advancements toward the ultimate goal of achieving a biomolecular computer. It shows that with proper coding and use of appropriate enzyme, a computational device is possible without any outside mediation. This opens up the door for using such devices not just for performing computation, but potentially also in genetics and medicine.

7 Concluding Remarks

There are many other issues concerning biomolecular computations that are rather significant for a prosperous development of the scientific discovery that were not included in this short review. They are both theoretical and experimental. In use of synthetic DNA one of the difficult questions is encoding of the bases such that cross hybridization of non-complementary strands is minimized. It turned out that this problem is quite complex and many authors have concentrated on developing theoretical coding models[73,74], computer simulations[75–77] even experimental built up of coding libraries[78].

Fig.10. Several computational steps of the finite state machine.

From the theoretical point of view, although there are many theoretical models for DNA-based computations, the real task in front of the theoreticians is to characterize these models within the scope of the experimental limitations.

Besides the Adleman's latest success[39], all other experimental results in biomolecular computing have been still on a "toy" level. Although Adleman's experiment solved a computational complex problem (an NP-complete problem), it is now clear that due to the amount of DNA needed to scale this approach to a larger problem, solutions to large combinatorial search problems by biomolecular protocols will not improve on the conventional computers. Even more, there are real challenges for the experimentalists to obtain protocols that are sufficiently reliable, controllable and predictable. The standard biomolecular protocols do not have the precision needed for computation. These techniques have to go through many adjustments and sometimes completely new protocols are necessary in order to improve their yield. The solution of a 20-variable SAT problem was obtained by designing protocols for exquisitely sensitive and error-resistant separation of small set of molecules. The solution of a 20-variable SAT problem could be considered as a "test" for this new protocol. Hence, a search for a DNA solution of such combinatorial problems, even though not computationally significant, may prove to be fruitful in developing new technologies.

On the other side, the whole research area has been very recent and for such a small period (less than 9 years since the first experimental result) the progress has been tremendous. For the past half-century DNA, RNA and proteins have been ex-

clusively the provence of molecular biologists and medical scientists who have concentrated on understanding their biological impact and properties in living organisms. As these past few years of biomolecular computing has shown, it is most likely that nanoengineers, computer scientists and material scientists will explore these molecules within non-biological contexts. We now have models and experimental designs for two dimensional arrays, three dimensional structures, DNA devices fueled by DNA strands, autonomous finite state biomolecular machines... All of these are in an infantile stage and it is not clear which one of these several roads will turn out to be fruitful. It is a real possibility that quite few of them may be successful, each in a different field of our scientific community and aspects of life. Whatever the results, this area of research has brought together theoreticians (mathematicians and computer scientists) with experimentalists (molecular biologists and biochemists) to one very successful collaboration. Just the exchange of fresh ideas and discussions among these communities brings excitement, and quite often, provides a new line of development that could not have been possible without the "outsiders".

References

[1] Head T. Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors. *Bull. Math. Biology,* 1987, 49: 737–759.

[2] Adleman L. Molecular computation of solutions of combinatorial problems. *Science*, 1994, 266: 1021–1024.

[3] Lipton R, Baum E (Eds.) DNA based computers. In *Proc. the First Annual Meeting, DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* 27 Providence RI, American Mathematical Society, 1996.

[4] Landweber L, Baum E (Eds.). DNA based computers. In *Proc. the First Annual Meeting, DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* 44 Providence RI, American Mathematical Society, 1998.

[5] Rubin H, Wood D (Eds.). DNA based computers. In *Proc. the Third Annual Meeting DIMACS series in Discrete Math. and Theoretical Comp. Sci.,* Vol.48, 1999.

[6] Kari L, Wood D (Eds.). DNA based computers, revised papers. In *Proc. the Fourth Annual Meeting BioSystems* (special issue), Vol.52, 1999.

[7] Winfree E, Gifford D K (Eds.). DNA based computers. In *Proc. the Fifth Annual Meeting, DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* 54 Providence RI, American Mathematical Society, 2000.

[8] Condon A, Rozenberg G (Eds.). DNA based computers. In *Proc. the Sixth Annual Meeting,* LNCS 2054, Springer-Verlag, 2001.

[9] Jonoska N, Seeman N C (Eds.). DNA based computers. In *Proc. the Seventh Annual Meeting,* LNCS 2340, Springer-Verlag, 2002.

[10] Hagiya M, Ohuchi A (Eds.). DNA based computers. In *Proc. the Eighth Annual Meeting,* LNCS 2568, Springer-Verlag, 2002.

[11] Head T, Paun Gh, Pixton D. Language theory and molecular genetics. Handbook of Formal Languages, Vol.II, Rozenberg G, Salomaa A (Eds.), Springer Verlag, 1997, pp.295–358.

[12] Paun Gh, Rozenberg G, Salomaa A. DNA Computing, New Computing Paradigms. Springer Verlag, 1998.

[13] Kari L. DNA computing: Arrival of biological mathematics. *The Mathematical Intelligencer,* 1997, 19(2): 9–22.

[14] Ausubel F M, Brent R, Kingston R E *et al.* Current Protocols in Molecular Biology. Greene Publishing Associates and Wiley-Interscience, New York, 1993.

[15] Seeman N C. The design of single-stranded nucleic acid knots. *Molecular Engineering,* 1992, 2: 197–307.

[16] Seeman N C *et al.* The perils of polynucleotides: The experimental gap between the design and assembly of unusual DNA structures. in [4], pp.215–234.

[17] Seeman N C, Zhang Y, Du S M *et al.* Construction of DNA polyhedra and knots through symmetry minimization. *Supermolecular Stereochemistry,* Siegel J S (Ed.), 1995, pp.27–32.

[18] Du S M, Wang H, Tse-Dinh Y C, Seeman N C. Topological transformations of synthetic DNA knots. *Biochemistry,* 1995, 34: 673–682.

[19] Mao C, Sun W, Seeman N C. Construction of borromean rings from DNA. *Nature,* 1997, 386: 137–138.

[20] Chen J, Seeman N C. Synthesis from DNA of a molecule with the connectivity of a cube. *Nature,* 1991, 350: 631–633.

[21] Chen J, Kallenbach N R, Seeman N C. A specific quadrilateral synthesized from DNA branched junctions. *Journal of the American Chemical Society,* 1989, 111: 6402–6407.

[22] Mao C, Sun W, Seeman N C. Designed two-dimensional Holliday junction arrays visualised by atomic force microscopy. *Journal of the American Chemical Society,* 1999, 121: 5437–5443.

[23] Seeman N C *et al.* Gel electrophoretic analysis of DNA branched junctions. *Electrophoresis,* 1989, 10: 345–354.

[24] Seeman N C. Nucleic acid junctions and lattices. *Journal of Theoretical Biology,* 1982, 99: 237–247.

[25] Jonoska N, Sa-Ardyen P, Seeman N C. Computation by self-assembly of DNA graphs. *Journal of Genetic Programming and Evolvable Machines,* 2003, 4: 123–137.

[26] Winfree E, Liu F, Wenzler L, Seeman N C. Design of self-assembly of two-dimensional crystals. *Nature,* 1998, 494: 539–544.

[27] Seeman N C. Private Communication.

[28] LaBean T H, Yan H, Kopatsch J *et al.* The construction, analysis, ligation and self-assembly of DNA triple crossover complexes. *J. Am. Chem. Soc.,* 2000, 122: 1848–1860.

[29] Adleman L. On constructing a molecular computer. DNA Based Computers. *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science,* Lipton R, Baum E (Eds.), 27 Providence RI, American Mathematical Society, 1996, pp.1–21.

[30] Lipton R. DNA solution of hard computational problems. *Science*, 1995, 268: 542–545.

[31] Adleman L, Rothemund P W K, Roweis S *et al.* On applying molecular computation to the Data Encryption Standard. *DIMACS series in Discrete Math. and Theoretical Comp. Sci.,* 1999, 44: 31–44.

[32] Hartmanis J. On the weight of a computation. *Bul. EATCS,* 1995, 55: 136–138.

[33] Morimoto N, Arita M, Suyama A. Solid phase DNA solution to the Hamiltonian path problem. DNA Based Computers III, Rubin H, Wood D H (Eds.), *DIMACS Series AMS,* 1999, 48: 193–206.

[34] Ouyang Q, Kaplan P D, Liu S *et al.* DNA solution to the Maximal Clique problem. *Science*, 1997, 278: 446–449.

[35] Faulhammer D, Cukras A R, Lipton R J *et al.* Molecular computation: RNA solutions to chess problems. In *Proc. Natl. Acad. Sci. USA,* 2000, 97: 1385–1389.

[36] McCaskill J S, Penchovsky R, Gohlke M *et al.* Steady flow micro-reactor module for pipelined DNA computations. *DNA Computing,* Condon A, Rozenberg G (Eds.), LNCS 2054, Springer, 2001, pp.263–270.

[37] van Noort D, Gast F, McCaskill J S. DNA computing in microreactors. *DNA Computing,* Jonoska N, Seeman N C (Eds.), LNCS 2340, Springer, 2002, pp.31–43.

[38] Penchovsky R, McCaskill J S. Cascadable hybridisation transfer of specific DNA between microreactor selection modules. *DNA Computing,* Jonoska N, Seeman N C (Eds.), LNCS 2340, Springer, 2002, pp.44–53.

[39] Braich R S, Chelyapov N, Johnson C *et al.* Solution of a 20-variable 3-SAT problem on a DNA computer. *Science,* 2002, 296: 499–502.

[40] Suyama A. Programmable DNA computer. In *Preliminary Proceedings of the 8th Int. Meeting on DNA Based Computers,* Hokkaido University, June 10–13, 2002, p.91.

[41] Sakamoto K, Gouzu H, Komiya K, Kiga D, Yokoyama S, Yokoyama T, Yokomori M. Molecular computation by DNA hairpin formation. *Science,* 2000, 288: 1223–1226.

[42] Hagiya M, Arite M, Kiga D *et al.* Towards parallel evaluation and learning of Boolean μ-formulas with molecules. DNA Based Computers III, Rubin H, Wood D H (Eds.), *DIMACS Series AMS,* 1999, 48: 57–72.

[43] Winfree E. Whiplash PCR for $O(1)$ computing. In *Preliminary Proceedings of the 4th International Meeting on DNA Based Computers,* Kari L, Rubin H, Wood D H (Eds.), University of Pennsylvania, June 15–19, 1998, pp.175–188.

[44] Winfree E, Yang X, Seeman N C. Universal computation via self-assembly of DNA: Some theory and experiments. in [3].

[45] Wang H. Notes on a class of tiling problems. *Fundamenta Mathematicae,* 1975, 82: 295–334.

[46] Rothemund P, Papadakis N, Winfree E. Algorithmic self-assembly of DNA Sierpinsky triangles. In *Preliminary Proceedings of the 9th International Meeting on DNA Based Computers,* Chen J, Reif J (Eds.), Madison, Wisconsin, June 1–4, 2003, p.125.

[47] Winfree E, Eng T, Rozenberg G. String tile models for DNA computing by self-assembly. *DNA Computers,* Condon A, Rozenberg G (Eds.), LNCS 2054, Springer, 2001, pp.63–88.

[48] Mao C, LaBean T, Reif J H *et al.* Logical computation using algorithmic self-assembly of DNA triple crossover molecules. *Nature,* 2000, 407: 493–496.

[49] Jonoska N, Karl S, Saito M. Three dimensional DNA structures in computing. *BioSystems,* 1999, 52: 143–153.

[50] Jonoska N, Karl S, Saito M. Creating 3-dimensional graph structures with DNA. in [4], pp.123–136.

[51] Wasserman S, Cozzarelli N. Biochemical topology: Applications to DNA recombination and replication. *Science,* 1986, 232: 951–960.

[52] Jonoska N, Saito M. Boundary components of thickened graphs. In *Revised Papers of 7th International Meeting on DNA Based Computers,* Jonoska N, Seeman N C (Eds.), LNCS 2340, Springer-Verlag, 2002, pp.70–81.

[53] Head T. Splicing schemes and DNA. In *Lindenmayer Systems, Impacts on Theoretical Computer Science and Developmental Biology,* Springer-Verlag, 1992, pp.371–383.

[54] Paun Gh. On the power of the splicing operation. *Int. J. Computer Math,* 1995, 59: 27–35.

[55] Paun Gh, Rozenberg G, Salomaa A. Computing by splicing. *Theoretical Computer Science,* 1996, 168(2): 321–336.

[56] Ehrenfeucht A, Rozenberg G. Forbidding-enforcing systems. *Theoretical Computer Science,* 2003, 292: 611–638.

[57] van Vugt N. Models of molecular computing [Dissertation]. Leiden University, May 2002.

[58] Filipov D, Jonoska N. The topological space of formal languages and fe-systems. manuscript.

[59] Cavaliere M, Jonoska N. Forbidding and enforcing in membrane computing. *J. Natural Computing,* 2003, 2: 215–228.

[60] Roweis S, Winfree E, Burgoyne R *et al.* A sticker based architecture for DNA computation. in [4], pp.1–27.

[61] Takahara A, Yokomori T. On the computational power of insertion-deletion systems. in [10], pp.269–280.

[62] Freund R, Păun G, Rozenberg G, Salomaa A. Watson-Crick automata. Tech. Report 97-13, Dept. of Computer Science, Leiden Univ. 1997.

[63] Head T, Pixton D, Goode E. Splicing systems: Regularity and below. in [10], pp.262–268.

[64] Schützenberger M P. Sur Certaines Opérations de fermeture dans les language rationnels. *Symposia Mathematica*, 1975, 15: 245–253. (Also in *RAIRO Information Theory*, 1974, 1: 55–61.)

[65] de Luca A, Restivo A. A characterization of strictly locally testable languages and its applications to subsemigroups of a free semigroup. *Information and Control*, 1980, 44: 300–319.

[66] Yurke B, Turberfield A J, Mills A P *et al.* A DNA fueled molecular machine made of DNA. *Nature,* 2000, 406: 605–608.

[67] Simmel F C, Yurke B. Operation of a purified DNA nanoactuator. in [9], pp.248–257.

[68] Simmel F C, Yurke B. Using DNA to construct and power a nanoactuator. *Phys. Rev. E.,* 2001, 63(041913): 1–5.

[69] Yan H, Zhang X, Shen Z, Seeman N C. A robust DNA mechanical device controlled by hybridization topology. *Nature,* 2002, 415: 62–65.

[70] Seeman N C. DNA in a material world. *Nature,* 2003, 421(6921): 427–431.

[71] Benenson Y, Paz-elizur T, Adar R *et al.* Programmable and autonomous machine made of biomolecules. *Nature,* 2001, 414: 430–434.

[72] Benenson Y, Adar R, Paz-elizur T *et al.* Molecular computing machine uses information as fuel. In *Preliminary Proceedings of the 8th Int. Meeting on DNA Based Computers,* Hokkaido University, June 10–13, 2002, p.198.

[73] Hussini S, Kari L, Konstantinidis S. Coding properties of DNA languages. *DNA Computing: Proceedings of the 7th International Meeting on DNA Based Computers*, Jonoska N, Seeman N C (Eds.), LNCS 2340, Springer, 2002, pp.57–69.

[74] Mahalingam K, Jonoska N. Languages of DNA based code words. In *Preliminary Proceedings of the 9th International Meeting on DNA Based Computers,* Chen J, Reif J, (Eds.), Madison, Wisconsin, June 1–4, 2003, pp.58–68.

[75] Garzon M, Deaton R, Reanult D. Virtual test tubes: A new methodology for computing. In *Proc. 7th. Int. Symp. String Processing and Information retrieval,* A Coruña, Spain. IEEE Computing Society Press, 2000, pp.116–121.

[76] Seeman N C. De Novo design of sequences for nucleic acid structural engineering. *J. of Biomolecular Structure & Dynamics,* 1990, 8(3): 573–581.

[77] Feldkamp U, Saghafi S, Rauhe H. DNASequenceGenerator – A program for the construction of DNA sequences. *DNA Computing*, Jonoska N, Seeman N C (Eds.), LNCS 2340, Springer, 2002, pp.23–32.

[78] Deaton R, Chen J, Bi H *et al.* A PCR-based protocol for In Vitro selection of non-crosshybridizing oligonucleotides. in [10], pp.196–204.

Nataša Jonoska received the B.S. degree in mathematics and computer science in 1984 from Univerity of Cyril & Methodius in Skopje, Macedonia. She obtained a Ph.D. degree in mathematical sciences from the State University of New York in Binghamton in 1993 and joined the faculty at the University of South Florida (USF) the same year. Now she is an associate professor in mathematics at USF and her research includes formal language theory, biomolecular computation, and symbolic dynamics. She organized the 7th International Meeting on DNA Based Computers and has served on the Program Committee of this conference for several years.