ELSEVIER

Theoretical Computer Science 231 (2000) 193–203

Theoretical Computer Science

www.elsevier.com/locate/tcs

Using DNA to solve the Bounded Post Correspondence Problem

Lila Kari[a,*], Greg Gloor[b], Sheng Yu[a]

[a] *Department of Computer Science, University of Western Ontario, London, Ont., Canada N6A 5B7*
[b] *Department of Biochemistry, University of Western Ontario, London, Ont., Canada N6A 5C1*

Abstract

Theoretical research in DNA computing includes designing practical experiments for solving various computational problems by means of DNA manipulation. This paper proposes a DNA algorithm for an NP-complete problem, The Bounded Post Correspondence Problem. The proposed experiment can be used to test several standard molecular biology laboratory procedures for their usability as bio-operations in DNA computing.

Keywords: DNA computing: biomolecular computing; NP-complete problem; Post correspondence problem

1. Introduction

Molecular computing, known also under the name of biomolecular computing, biocomputing or DNA computing, is a new computation paradigm that employs (bio)molecule manipulation to solve computational problems. The excitement generated by the first successful experiment [1] was due to the fact that computing with biomolecules (mainly DNA) offered an entirely new way of performing and looking at computations: the main idea was that data could be encoded in DNA strands, and molecular biology techniques could be used to execute computational operations. Besides the novelty of the approach, molecular computing has the potential to outperform electronic computers. For example, DNA computing has the potential to provide huge memories: DNA in weak solution in 1 l of water can encode 10^{19} bytes, and one can perform massively parallel associative searches on these memories [6, 42]. Computing with DNA also has the potential to supply massive computational power. A general proposed use of molecular computing is to construct parallel machines where

* Corresponding author.
E-mail addresses: lila@csd.uwo.ca, http://www.csd.uwo.ca/~lila (L. Kari), syu@csd.uwo.ca (S. Yu), ggloor@julian.uwo.ca (G. Gloor)

0304-3975/00/$ - see front matter
PII: S0304-3975(99)00100-0

each processor's state is encoded by a DNA strand. DNA in weak solution in 1 l of water can encode the state of 10^{18} processors. Moreover, one can perform massively parallel computations by executing recombinant bio-operations that act on all the DNA molecules at the same time. These recombinant bio-operations may be used to execute massively parallel memory read/write, logical operations and also further basic operations, such as parallel arithmetic. Since certain recombinant bio-operations can take minutes to perform, the overall potential for molecular computation is about 1000 tera-ops [42].

Despite the progress obtained, substantial obstacles remain before molecular computing becomes an effective computational paradigm. The field is therefore still in the incipient stage of (i) testing the suitability of certain molecular biology techniques for computational purposes, and (ii) finding a suitable formal model for DNA computing. The research in the field has had therefore, from the beginning, both experimental and theoretical aspects. (For surveys and summaries of the field see [28] and references therein, see also [42, 18, 45, 49].)

The experiments that have actually been carried out include the following. Kaplan et al. [26], replicated Adleman's experiment; a Wisconsin team of computer scientists and biochemists made partial progress in solving a 5-variable instance of the SAT problem by using a surface-based approach [36]; Guarnieri, Fliss and Bancroft have used a horizontal chain-reaction for DNA-based addition [19]. At the same time, various aspects of the implementability of DNA computing have been experimentally investigated: the effect of good encodings on the solutions of Adleman's problem was addressed in [14]; the complications raised by using the polymerase chain reaction were studied in [27]; the usability of self-assembly of DNA was studied in [55]; the experimental gap between the design and assembly of unusual DNA structures was pointed out in [48]; joining and rotating data with molecules was reported in [4]; concatenation with PCR was studied in [4, 5]; evaluating simple Boolean formulas was started in [20]; ligation experiments in computing with DNA were conducted in [25]; an experiment featuring a DNA solution to the shortest common superstring problem is currently underway at Western Ontario [29].

The theoretical work on DNA computing comprises, on one side, attempts to model the process in general, and to give it a mathematical foundation. To this aim, models of DNA computing have been proposed and studied from the point of view of their computational power and their in vitro feasibility (see, for example, [2, 3, 8, 21, 30, 35, 54, 50, 22, 40, 39, 16, 13, 51, 31, 9, 11, 41, 43, 44, 53]). Overall, the existence of different models with complementing features shows the versatility of DNA computing and increases the likelihood of practically constructing a DNA computing-based device.

On the other side, the theoretical research on DNA computing includes designing potential experiments for solving various problems by means of DNA manipulation. Descriptions of such experiments include the satisfiability problem [35], breaking the data encryption standard [10], expansions of symbolic determinants [34], matrix multiplication [38], graph connectivity and knapsack problem using dynamic programming

[7], road coloring problem [24], computer algebra problems [52], and simple Horn clause computation [33]. The present paper falls into this category by proposing a DNA algoritm for an NP-complete problem, The Bounded Post Correspondence Problem, as a test-bed for several bio-operations.

It is anticipated that the research in molecular computing will have a great impact in many aspects of science and technology. In particular, molecular computing sheds new light onto the very nature of computation, while also opening prospects of computing devices radically different from today's computers. Probing the limits of biomolecular computation could lend new insights into the information processing abilities of cellular organisms and in general into computational processes in nature.

2. An NP-complete problem

The problem we have chosen for our experiment is *The Bounded Post Correspondence Problem*. There were several reasons for our choice. First, the problem is NP-complete, i.e., it is a *hard* computational problem. This means, in particular, that it cannot yet be solved in real-time by electronic computers. Finding an efficient DNA algorithm for solving it would thus indicate that DNA computing could be quantitatively superior to electronic computing. Second, the experiment proposed for solving the problem uses readily available reagents and techniques. In fact, one of the purposes of the experiment is to test standard molecular procedures for potential use in DNA computing. Last, but not least, the Bounded Post Correspondence Problem is a much celebrated computer science problem. If the condition "bounded" were dropped, the resulting problem would be *unsolvable* by classical means of computation. The search for DNA solutions of this problem could thus give insights into the limitations of DNA computing, and shed light into the conjecture that DNA computing is a qualitatively new model of computation.

Before formally stating the problem, we summarize the notation used throughout the paper. For a set Σ, card(Σ) denotes its cardinality, that is, the number of elements in Σ. An *alphabet* is a finite nonempty set. Its elements are called *letters* or *symbols*. The symbols will be usually denoted by the first letters of the alphabet, with or without indices, i.e., a, b, C, D, a_i, b_j, etc. If $\Sigma = \{a_1, a_2, \ldots, a_n\}$ is an alphabet, then any sequence $w = a_{i_1} a_{i_2} \ldots a_{i_k}$, $k \geqslant 0$, $a_{i_j} \in \Sigma$, $1 \leqslant j \leqslant k$ is called a *string* (*word*) over Σ. The length of the word w is denoted by $|w|$ and, by definition, equals k. The words over Σ will usually be denoted by the last letters of the alphabet, with or without indices, for example x, y, w_j, u_i, etc. The set of all words consisting of letters from Σ will be denoted by Σ^*. For additional formal language definitions and notations see [46].

2.1. Bounded Post Correspondence Problem [17,12]

INSTANCE: Alphabet Σ, two lists of strings from Σ^*, $u = (u_1, u_2, \ldots, u_n)$ and $w = (w_1, w_2, \ldots, w_n)$, and a positive integer $K \leqslant n$.

QUESTION: Is there a sequence $i_1, i_2, \ldots, i_k$ of $k \leqslant K$ (not necessarily distinct) positive integers, each between 1 and n, such that the two strings $u_{i_1}u_{i_2}\ldots u_{i_k}$ and $w_{i_1}w_{i_2}\ldots w_{i_k}$ are identical?

Comments: Problem is undecidable if no upper bound is placed on k [23].

In order to be able to state the problem in molecular biology terms and give it a DNA-based solution, we need a brief introduction of some basic molecular biology notions. For further details of molecular biology terminology, the reader is referred to [32].

DNA (deoxyribonucleic acid) is found in every cellular organism as the storage medium for genetic information. It is composed of units called nucleotides, distinguished by the chemical group, or base, attached to them. The four bases are *adenine*, *guanine*, *cytosine* and *thymine*, abbreviated as A, G, C, and T. (The names of the bases are also commonly used to refer to the nucleotides that contain them.) Single nucleotides are linked together end-to-end to form DNA strands. A short single-stranded polynucleotide chain, usually less than 30 nucleotides long, is called an *oligonucleotide*. The DNA sequence has a *polarity*: a sequence of DNA is distinct from its reverse. The two distinct ends of a DNA sequence are known under the name of the $5'$ end and the $3'$ end, respectively. Taken as pairs, the nucleotides A and T and the nucleotides C and G are said to be *complementary*. Two complementary single-stranded DNA sequences with opposite polarity will join together to form a double helix in a process called *base-pairing* or *annealing*. The reverse process – a double helix coming apart to yield its two constituent single strands – is called *melting*.

A single strand of DNA can be likened to a string consisting of a combination of four different symbols, A, G, C, T. Mathematically, this means we have at our disposal a 4-letter alphabet $\Sigma = \{A, G, C, T\}$ to encode information. As concerning the operations that can be performed on DNA strands, the proposed models of DNA computation are based on various combinations of the following primitive *bio-operations* [28, 29]:

- *Synthesizing* a desired polynomial-length strand.
- *Mixing*: pour the contents of two test-tubes into a third.
- *Annealing* (*hybridization*): bond together two single-stranded complementary DNA sequences by cooling the solution.
- *Melting* (*denaturation*): break apart a double-stranded DNA into its single-stranded components by heating the solution.
- *Amplifying* (*copying*): make copies of DNA strands by using the polymerase chain reaction (PCR) [15].
- *Separating* the strands by length using a technique called gel electrophoresis.
- *Extracting* those strands that contain a given pattern as a substring by using affinity purification.
- *Cutting* DNA double-strands at specific sites by using commercially available restriction enzymes.
- *Ligating:* paste DNA strands with compatible sticky ends by using DNA ligases.

- *Substituting:* substitute, insert or delete DNA sequences by using PCR site-specific oligonucleotide mutagenesis.
- *Detecting and Reading* a DNA sequence from a solution.

We are now ready to formulate the Bounded Post Correspondence Problem in molecular biology terms:

2.2. The Bounded Post Correspondence Problem in molecular terms

Consider two lists of n oligonucleotide sequences each, $u=(u_1,u_2,\ldots,u_n)$ and $w=(w_1,w_2,\ldots,w_n)$. Given a number K, less than or equal to n, are there two sequences of catenated oligonucleotide strings with the properties:

- one sequence contains only oligonucleotide strings from the list u, and the other only oligonucleotide strings from the list w,
- each sequence contains the same number (smaller than K) of oligonucleotide strings,
- the oligonucleotide strings are catenated in the same order,
- the two sequences obtained from catenating are identical?

Note that the given oligo strings are not necessarily of the same length, that they are are not necessarily distinct, and that in the joined sequences oligo strings can be repeated. For example, let $u=(TAT,GTAA,A,AT)$, $w=(TA,CAGG,TA,GC)$ and $K=4$. The oligonucleotide sequences in the u-list are not of the same length, and in the list w, the 1st sequence coincides with the 3rd. Moreover, the answer to our problem is "YES". Indeed the sequence *TATA* which is the catenation of the 1st and 3rd oligo-strings from the list u coincides with the sequence obtained by catenating the 1st and 3rd oligo-strings from the list w. Notice that also the sequence *TATATATA* obtained by catenating the 1st, 3rd, 1st and 3rd strings from the list u coincides with the corresponding sequence from the list w.

3. Molecular solution

This section contains a DNA algorithm developed to solve the Bounded Post Correspondence Problem, and which we propose as a practical experiment. The experiment uses readily available reagents and techniques. Compared to standard protocols, the main difference is the number of reactions conducted entirely in vitro prior to cloning of the products. Careful optimization of each step will be required to ensure maximum specificity. However, these reactions are possible to perform entirely within the established parameters of each enzyme. All reactions will be performed on nucleic acids bound to a matrix. This will simplify recovery of the desired products and will permit reagents and buffers to be pumped through in bulk rather than be added individually. This solid-phase approach will reduce the number of manipulations and provide maximum control over the reaction conditions.

DNA algorithm for the Bounded Post Correspondence Problem

Step 1. *Encoding the words and numbers.*

(a) Encode each word u_i, w_i, $1 \leqslant i \leqslant n$, in a DNA sequence.

(b) Encode each number i, $1 \leqslant i \leqslant n$, in a DNA sequence.

(c) Synthesize the encoding of a chosen "bridge" string β into a DNA sequence and put the resulting population of sequences β into two tubes A and B.

(d) Set $k = 1$.

Step 2. *Generating catenated oligonucleotide sequences from both lists.*

(1) Distribute the contents of tube A into n tubes $A_1, A_2, \ldots, A_n$.

(2) For each tube A_i, append the encoding of the word u_i at the beginning of all strings in A_i, and the encoding of the number i at the end of all strings in A_i.

(3) Mix the contents of the resulting tubes A_i, $1 \leqslant i \leqslant n$, into the tube A. As a result of (1)–(3) tube A will contain all the possible combinations

$$u_{i_1} u_{i_2} \ldots u_{i_k} \ \beta \ i_k \ldots i_2 i_1, \quad 1 \leqslant i_j \leqslant n, \ \ 1 \leqslant j \leqslant k.$$

(4) Distribute the contents of tube B into n tubes $B_1, B_2, \ldots, B_n$.

(5) For each tube B_i, append the encoding of the word w_i at the beginning of all strings in B_i, and the encoding of the number i at the end of all strings in B_i.

(6) Mix the contents of the resulting tubes B_i, $1 \leqslant i \leqslant n$, into the tube B. As a result of (4), (5) and (6), tube B will contain all the possible combinations

$$w_{i_1} w_{i_2} \ldots w_{i_k} \ \beta \ i_k \ldots i_2 i_1, \quad 1 \leqslant i_j \leqslant n, \ \ 1 \leqslant j \leqslant k.$$

Step 3. *Search for matching catenated sequences.*

Check if the tubes A and B contain any identical strings. If yes, the answer to our problem is YES. Otherwise, i.e., if no matching string has been found, increase the value of k by 1.

Step 4. If $k > K$ then stop and say NO. Otherwise go to *Step 2*.

Implementation (see Appendix A)

The problem can be solved by first separately making all the possible ordered combinations of concatenations containing k words and the k corresponding indices from the sequence lists u and w. To check if any identical strings have been obtained we propose the following method. The pools of concatenated sequences will be hybridized, and treated with a single-stranded nuclease to degrade any single-stranded DNA, while leaving the double-stranded DNA intact. Only the u-strings and w-strings that are complementary, of the same length, and in the same order will survive this treatment. They will be then amplified by the PCR, cloned and sequenced to find the answer. In principle, this problem is similar to a subtractive hybridization [37, 47].

Step 1. *Encoding the words and numbers.*

(a) Encoding the words: Each oligonucleotide encoding a word, and its complement will be synthesized. One strand will be synthesized unphosphorylated on the 5′ end,

and with a 3′ hydroxyl group at the other end. The complementary oligonucleotide strand will be synthesized with a 5′ phosphate group, and will be blocked with an amino group on the 3′ end. In this way, only one end of the double-stranded DNA sequence that arises by annealing of the oligonucleotides is competent to be ligated by T4 DNA ligase. The words could have a minimum size of 1 nucleotide and will be encoded as sequences adjacent to the right side of the recognition site for a restriction endonuclease that makes a cut outside its recognition sequence. The upper limit on the word length is constrained only by the length of sequence that can be synthesized efficiently. Currently, this upper limit is a chain length of about 100 nucleotides. (Note that a variety of restriction endonucleases will be assessed for their suitability for this reaction).

(b) *Encoding the numbers:* Each oligonucleotide that represents a number, and its complement will be synthesized as a 12-base long oligonucleotide. The oligos will have a cut site for a blunt-cutting restriction endonuclease in them that is adjacent to the right side of the sequence encoding the number. As described above, the oligonucleotides will be synthesized so that only one end of the double-stranded sequence will be competent for ligation.

(c) Synthesize the encoding of a chosen "bridge" string β into a DNA sequence. Put the resulting population of sequences β into two tubes A and B. The "bridge" will be biotinylated at one or more positions and bound to an avidin afinity column. In this way a continuous flow system can be set up to better remove unwanted products after the reaction. The bridge for the words in the u-list will be biotinylated on the top strand and the bridge for the words in the w-list will be biotinylated on the bottom strand. This will allow separation of the top and bottom strands in later stages.

(d) Set $k = 1$.

Step 2. *Generating catenated oligonucleotide sequences from both lists.*

(1) *Distribute* the contents of tube A into n tubes $A_1, A_2, \ldots, A_n$.

(2) *Concatenating the words and corresponding numbers.*

It is important that the strings be joined efficiently and with a minimum of side reactions. This can proceed through sequential DNA ligation and endonuclease cleavage reactions as seen in the attached Appendix A.

In the beginning, in each tube, the word nucleotide will be joined to the left, and the number oligonucleotide will be joined to the right of the central "bridge" oligonucleotide. For example, the reactions used in tube A_1, to add the word u_1 to the left of the "bridge" and the number 1 to the right of it, will proceed as follows:

- *Ligation* of the word u_1 to the bridge oligonucleotide. The word will be annealed to its complement, and ligated to the left of the bridge. In this and all subsequent ligations the new oligonucleotide will be in vast excess (>100 fold) to prevent self-ligation of the concatenated product.
- *Cleavage* (*cutting*) on the right (number) side to expose a ligatable end with a blunt-cutting endonuclease.
- *Ligation* of the oligonucleotide representing number 1 to the exposed side.

- *Cleavage* on the left (word) side to expose a ligatable end. This cleavage will be done with a restriction endonuclease that cleaves outside the recognition site. In this way the word sequences can be arbitrarily chosen.
- *Filling in* of the cleaved end by a DNA polymerase and the four dNTP's to generate a blunt end.

Reactions similar to the ones above will be performed in separate tubes for each of the given n words. As a result, each tube A_i will contain the ith word concatenated to the left of the bridge and the number i concatenated to its right.

(3) *Mixing* of the products from the n tubes obtained in (2) in the tube A.

(4)–(6) Use a method analogous to the one in (1)–(3) for generating catenated oligo sequences of words from the w-list and the corresponding numbers indicating their indices.

Step 3. *Search for matching catenated sequences.*

Take aside small portions of the products from tubes A and B and mix them. Expose each end in turn and ligate a unique sequence onto each end for PCR amplification. Separate the top and bottom strands, keeping only the top strands for the sequences from the u-list and the bottom strands for the sequences from the w-list. The selection of complementary products from the two populations of concatenated sequences of words from the u-list, respectively, from the w-list can be done as follows: perform a simple hybridization followed by digestion with a single-strand specific nuclease. The nuclease will degrade any single-stranded regions of DNA that remain. After the nuclease is removed, only those products that are fully double-stranded will remain.

Complementary products that remain will be amplified by PCR between the terminal given sequences. PCR products will be sequenced to verify that they can be derived from the addition reactions done. Verified products will indicate that the answer is YES, and our experiment is complete.

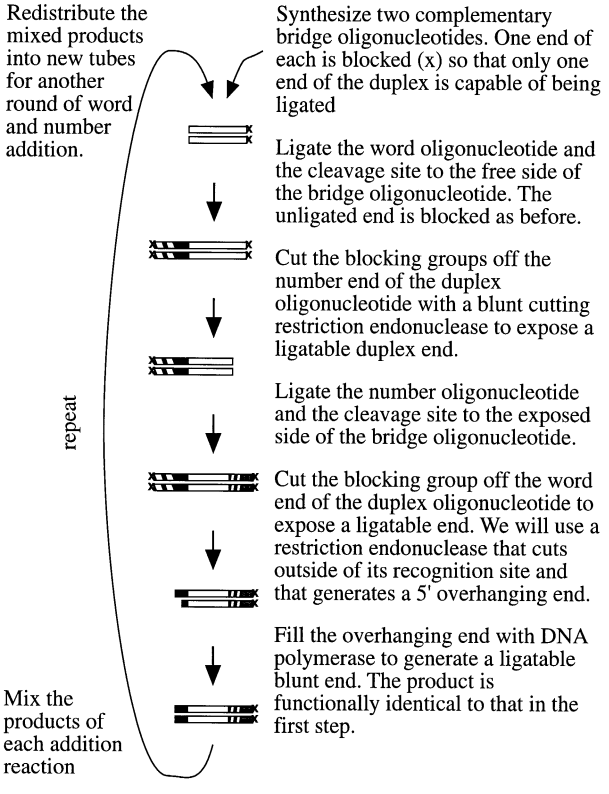
The absence of product indicates that no matching pairs have yet been found and we have to increase the value of k by one and continue the process.

Step 4. If $k > K$, the absence of product (identical catenated sequences formed by at most K words) actually indicates that the answer to our problem is NO and the experiment is complete.

If $k < K$, then take the tubes A and B with their remaining products and repeat the procedure starting from *Step* 2. Subsequent additions, one at a time, will be to both ends of the initial word-bridge-number oligonucleotide sequence: the word oligos to the left and the number oligos to the right of it. Each of the words will be added in a different tube or column to ensure that each word is represented at each position in the concatenated product. With sufficient chromatography equipment, this can be done in parallel.

Pilot experiments: Experiments can be set up working from a known positive as well as a negative result to verify that the reactions are working as described above. Controls will be done to ensure that single-base mismatches can be cleared by the chosen nuclease. It is likely that we will have to test several nucleases for the required activity and specificity.

Appendix A



References

[1] L. Adleman, Molecular computation of solutions to combinatorial problems, Science 266 (1994) 1021–1024.
[2] L. Adleman, On constructing a molecular computer, 1st DIMACS Workshop on DNA Based Computers, Princeton, 1995, DIMACS Series, Vol. 27, AMS Press, 1996, pp. 1–21.
[3] M. Amos, A. Gibbons, D. Hodgson, Error-resistant implementation of DNA computation, 2nd DIMACS Workshop on DNA Based Computers, Princeton, 1996, DIMACS Series, Vol. 44, 1999, AMS Press, pp. 151–161.
[4] M. Arita, M. Hagiya, A. Suyama, Joining and rotating data with molecules, Proc. 1997 IEEE Internat. Conf. on Evolutionary Computation, IN, pp. 243–248.
[5] M. Arita, A. Suyama, M. Hagiya, A heuristic approach for Hamiltonian Path Problem with molecules, Genetic Programming 1997: Proc. 2nd Ann. Conf. (GP-97), Morgan Kaufmann, Los Altos, CA, in press.
[6] E. Baum, Building an associative memory vastly larger than the brain, Science 268 (1995) 583–585.
[7] E. Baum, D. Boneh, Running dynamic programming algorithms on a DNA computer, 2nd DIMACS Workshop on DNA Based Computers, DIMACS Series, Vol. 44, 1999, AMS Press, pp. 77–85.
[8] D. Beaver, Computing with DNA, J. Comput. Biol. 2 (1) (1995) 1–8.
[9] D. Beaver, A universal molecular computer, 1st DIMACS Workshop on DNA Based Computers, Princeton, 1995, DIMACS Series, Vol. 27, 1996, AMS Press, pp. 29–36.

[10] D. Boneh, C. Dunworth, R. Lipton, Breaking DES using a molecular computer, 1st DIMACS Workshop on DNA Based Computers, Princeton, 1995, DIMACS Series, Vol. 27, 1996, AMS Press, pp. 37–65.
[11] D. Boneh, R. Lipton, C. Dunworth, J. Sgall, On the computational power of DNA, Discrete Appl. Math. 71 (1996) 79–94.
[12] R. Constable, H. Hunt, S. Sahni, On the computational complexity of scheme equivalence, Report No. 74-201, Dept. of Computer Science, Cornell University, Ithaca, NY, 1974.
[13] E. Csuhaj-Varju, R. Freund, L. Kari, G. Paun, DNA computing based on splicing: universality results. Proceedings of 1st Annual Pacific Symposium on Biocomputing, Hawaii, 1996, L. Hunter, T. Klein, eds., World Scientific Publ., Singapore, 1996, 179–190.
[14] R. Deaton, R. Murphy, J. Rose, M. Garzon, D. Franceschetti, S. Stevens, A DNA based implementation of an evolutionary search for good encodings for DNA computation, Proc. 1997 IEEE Int. Conf. on Evolutionary Computation, Indianapolis, pp. 267–271.
[15] C.W. Dieffenbach, G.S. Dveksler (Eds.), PCR Primer: A Laboratory Manual, Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York, 1995, pp. 581–621.
[16] R. Freund, L. Kari, G. Paun, DNA computing based on splicing: the existence of universal computers. Theory of Computing Systems 32 (1999) 69–112.
[17] M. Garey, D. Johnson, Computers and Intractability. A Guide to the Theory of NP-completeness, Freeman, San Francisco, CA, 1979.
[18] D.K. Gifford, On the path to computation with DNA, Science 266 (1994) 993–994.
[19] F. Guarnieri, M. Fliss, C. Bancroft, Making DNA add, Science 273 (1996) 220–223.
[20] M. Hagiya, M. Arita, Towards parallel evaluation and learning of Boolean μ-formulas with molecules, 3rd DIMACS Workshop on DNA Based Computers, Philadelphia, 1997, pp. 105–114.
[21] T. Head, Formal language theory and DNA: an analysis of the generative capacity of recombinant behaviors, Bull. Math. Biol. 49 (1987) 737–759.
[22] T. Head, G. Paun, D. Pixton, Language theory and genetics. Generative mechanisms suggested by DNA recombination, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Vol. 2, Springer, Berlin, 1996, pp. 295–360.
[23] J. Hopcroft, J. Ulmann, Formal Languages and their Relation to Automata, Addison-Wesley, Reading, MA, 1969.
[24] N. Jonoska, S. Karl, A molecular computation of the road coloring problem, 2nd DIMACS Workshop on DNA Based Computers, Princeton, 1996, DIMACS Series, Vol. 44, 1999, AMS Press, pp. 87–96.
[25] N. Jonoska, S. Karl, Ligation experiments in computing with DNA, Proc. 1997 IEEE Internat. Conf. on Evolutionary Computation, IN, pp. 261–266.
[26] P. Kaplan, G. Cecchi, A. Libchaber, Molecular computation: Adleman's experiment repeated, Technical Report, NEC Research Institute, 1995.
[27] P. Kaplan, G. Cecchi, A Libchaber, DNA-based molecular computation: template–template interactions in PCR, 2nd DIMACS Workshop on DNA Based Computers, Princeton, 1996, DIMACS Series, Vol. 44, 1999, AMS Press, pp. 97–104.
[28] L. Kari, DNA computing: arrival of biological mathematics, Math. Intell. 19 (2) (1997) 9–22.
[29] L. Kari, From Micro-Soft to Bio-Soft: computing with DNA, Proc. BCEC'97 (Bio-Computing and Emergent Computation) Skovde, Sweden, World Scientific, Singapore, pp. 146–164.
[30] L. Kari, G. Thierrin, Contextual insertions/deletions and computability, Inform. Comput. 131 (1) (1996) 47–61.
[31] L. Kari, G. Paun, G. Thierrin, S. Yu, At the crossroads of DNA computing and formal languages: characterizing recursively enumerable languages using insertion/deletion systems, 3rd DIMACS Workshop on DNA Based Computers, Philadelphia, 1997, DIMACS Series, Vol. 48, 1999, AMS Press, pp. 329–346.
[32] J. Kendrew et al. (Eds.), The Encyclopedia of Molecular Biology, Blackwell Science, Oxford, 1994.
[33] S. Kobayashi, T. Yokomori, G. Sampei, K. Mizobuchi, DNA implementation of simple Horn clause computation, Proc. IEEE Internat. Conf. on Evolutionary Computation, IN, 1997, pp. 213–217.
[34] T. Leete, M. Schwartz, R. Williams, D. Wood, J. Salem, H. Rubin, Massively parallel DNA computation: expansion of symbolic determinants, 2nd DIMACS Workshop on DNA Based Computers, Princeton, 1996, DIMACS Series, Vol. 44, 1999, AMS Press, pp. 45–58.
[35] R. Lipton, DNA solution of hard computational problems, Science 268 (1995) 542–545.

[36] Q. Liu, Z. Guo, A. Condon, R. Corn, M. Lagally, L. Smith, A surface-based approach to DNA computation, 2nd DIMACS Workshop on DNA Based Computers, Princeton, 1996, DIMACS Series, Vol. 44, 1999, AMS Press, pp. 123–132.
[37] Maniatis et al., Molecular Cloning: A Laboratory Manual, Cold Spring Harbor Press, New York, 1982.
[38] J. Oliver, Computation with DNA: matrix multiplication, 2nd DIMACS Workshop on DNA Based Computers, Princeton, 1996, DIMACS Series, Vol. 44. 1999, AMS Press, pp. 113–122.
[39] G. Paun, On the power of the splicing operation, Internat. J. Comput. Math. 59 (1995) 27–35.
[40] G. Paun, A. Salomaa, DNA computing based on the splicing operation, Math. Japonica 43 (3) (1996) 607–632.
[41] J. Reif, Parallel molecular computation: models and simulations, Proc. 7th Ann. ACM Symp. on Parallel Algorithms and Architectures (SPAA'95) Santa Barbara, CA, July 1995, pp. 213–223.
[42] J. Reif, Paradigms for Biomolecular Computation, First International Conference on Unconventional Models of Computation, Auckland, New Zealand, January 1998. Published in Unconventional Models of Computation, edited by C.S. Calude, J. Casti, and M.J. Dinneen, Springer Publishers, January 1998, pp. 72–93, 161.
[43] P. Rothemund, A DNA and restriction enzyme implementation of Turing machines, 1st DIMACS Workshop on DNA Based Computers, Princeton, 1995, DIMACS Series, Vol. 27, 1996, AMS Press, pp. 75–119.
[44] S. Roweis, E. Winfree, R. Burgoyne, N. Chelyapov, M. Goodman, P. Rothemund, L. Adleman, A Sticker based model for DNA computation, 2nd DIMACS Workshop on DNA Based Computers, Princeton, 1996, DIMACS Series, Vol. 44, 1999, AMS Press, pp. 1–29.
[45] H. Rubin, Looking for the DNA killer app, Nature 3 (1996) 656–658.
[46] A. Salomaa, Formal Languages, Academic Press, New York, 1973.
[47] Sambrook et al., Molecular Cloning: A Laboratory Manual, 2nd ed., 1989.
[48] N. Seeman et al., The perils of polynucleotides: The experimental gap between the design and assembly of unusual DNA structures, 2nd DIMACS Workshop on DNA Based Computers, Princeton, 1996, DIMACS Series, Vol. 44, 1999, AMS Press, pp. 215–233.
[49] W. Smith, DNA computers in vitro and in vivo, 1st DIMACS Workshop on DNA Based Computers, Princeton, 1995, DIMACS Series, Vol. 27, 1996, AMS Press, pp. 121–185.
[50] T. Yokomori, S. Kobayashi, DNA-EC: a model of DNA computing based on equality checking, 3rd DIMACS Workshop on DNA Based Computers, Philadelphia, 1997, DIMACS Series, Vol. 48, 1999, AMS Press, pp. 347–360.
[51] T. Yokomori, S. Kobayashi, C. Ferretti, On the power of circular splicing systems and DNA computability, Proc. 1997 IEEE Internat. Conf. on Evolutionary Computation, IN, pp. 219–224.
[52] R. Williams, D. Wood, Exascale computer algebra problems interconnect with molecular reactions and complexity theory, 2nd DIMACS Workshop on DNA Based Computers, Princeton, 1996, pp. 260–268.
[53] E. Winfree, Complexity of restricted and unrestricted models of molecular computation, 1st DIMACS Workshop on DNA Based Computers, Princeton, 1995, DIMACS Series, Vol. 27, 1996, AMS Press, pp. 187–198.
[54] E. Winfree, On the computational power of DNA annealing and ligation, 1st DIMACS Workshop on DNA Based Computers, Princeton, 1995, DIMACS Series, Vol. 27, 1996, AMS Press, pp. 199–221.
[55] E. Winfree, X. Yang, N. Seeman, Universal computation via self-assembly of DNA: some theory and experiments. 2nd DIMACS Workshop on DNA Based Computers, Princeton, 1996, DIMACS Series, Vol. 44, 1999, AMS Press, pp. 191–213.