

# Model Checking Computational Tree Logic Using Sticker Automata

Weijun Zhu<sup>1(✉)</sup>, Yanfeng Wang<sup>2</sup>, Qinglei Zhou<sup>1</sup>, and Kai Nie<sup>1</sup>

<sup>1</sup> School of Information Engineering, Zhengzhou University,  
Zhengzhou 450001, Henan, China  
[zhuweijun@zzu.edu.cn](mailto:zhuweijun@zzu.edu.cn)

<sup>2</sup> College of Electric Information Engineering,  
Zhengzhou University of Light Industry, Zhengzhou 450002, China

**Abstract.** The molecular computing has been successfully employed to solve more and more complex computation problems. However, as an important complex problem, the Computational Tree Logic (CTL) model checking is still far from resolved under the circumstance of molecular computing, since it is still a lack of method. To address this issue, an autonomous model checking method is presented for checking all the basic constructs of CTL using DNA computing and sticker automata. As a result, the CTL model checking problem under the circumstance of molecular computing is solved preliminary. The simulated experimental results demonstrate the effectiveness of the new method in molecular biology.

**Keywords:** Model checking · Computational tree logic · DNA computing · Sticker automata

## 1 Introduction

Differ from an electronic computer, a DNA computer use DNA molecules as the carrier of computation. In 1994, a Turing Award winner professor Adleman solved a small scale Hamilton path problem with a DNA experiment [5], and it is considered the pioneer work in the field of DNA computing. Since the DNA computing has a huge advantage of parallel computing, this technique was subsequently developed rapidly. After this famous experiment, many models and methods based on DNA computing were presented for solving some complex computational problems, especially the famous NP-hard problems and PSPACE-hard ones. For examples, Lipton reported that promoted Adlemans idea and tried to solve the SAT problem [6], Ouyang presented a DNA-computing-based model for solving the maximal clique problem [7], Shapiro solved an automata problem of two states and two characters using the autonomous DNA computing technique [8].

One of the key differences between computer and other computing tools is the universality. Xu constructed a mathematical model called “probe machine” for the general DNA computer [10]. By integrating the storage system, operation system, detection system and control system into a whole, he gradually obtains

a real general DNA computer—“Zhongzhou DNA computer” [10]. According to result from [9], a probe machine is a nine-tuples consisting of data library, probe library, data controller, probe controller, probe operation, computing platform, detector, true solution storage and residue collector. It is an universal DNA computing model which can be realized in biology, and a Turing machine is just a “special case” of a probe machine [9]. This significant progress has raised the practical importance of the researches on DNA computing.

Beside the satisfiability problem, the Model Checking (MC) one is another important computational problem. And the two problems are correlative. The MC was proposed by the Turing Award winner Clarke et al. [1]. In order to describe the different temporal properties, the researchers have presented some different temporal logics. The Turing Award winner Pnueli introduced Linear Temporal Logic (LTL for short) into computer science in [2], and this logic can express the linear properties. The Turing Award winner Clarke proposed Computation Tree Logic (CTL for short) in [3, 4], and this logic can express the branch properties.

As a complex computational problem, the model checking under the circumstance of DNA computing is always the goal of researchers. In 2006, the Turing Award winner Prof. Emerson employed some DNA molecules to conduct CTL model checking for the first time [11]. As for LTL, the model checking is a PSPACE problem in the classical computing, and we have found a DNA-computing-based method, see Reference [12, 13], which can be used for checking all four basic constructs and some popular formulas. The basic constructs in CTL are:  $\text{EpUq}$ ,  $\text{ApUq}$ ,  $\text{EFp}$ ,  $\text{AFp}$ ,  $\text{EGp}$ ,  $\text{AGp}$ ,  $\text{EXp}$ ,  $\text{AXp}$ . We can obtain arbitrary CTL formula by combining these basic constructs recursively. Up to now, many basic constructs in CTL cannot be conducted model checking within the framework of DNA computing. This is the problem to be solved in this paper.

## 2 Preliminary

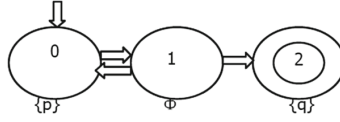
### 2.1 The Basic Constructs in CTL [1]

**Definition 1.** Let  $p$  and  $q$  be atomic propositions,  $\text{EpUq}$ ,  $\text{ApUq}$ ,  $\text{EFp}$ ,  $\text{AFp}$ ,  $\text{EGp}$ ,  $\text{AGp}$ ,  $\text{EXp}$  and  $\text{AXp}$  be the basic CTL construct. An arbitrary CTL formula can be obtained by combining recursively some basic CTL constructs. An atomic proposition and a basic CTL construct are interpreted on a system model  $M$ , and their intuitive meanings are given as follows.

- $p$  or  $q$  is satisfied in a state  $s$ , or not.
- $\text{EpUq}$  describes the following property: There exists at least one path in  $M$ , such that  $p$  is always satisfied until  $q$  is satisfied.
- $\text{ApUq}$  describes the following property: For each path in  $M$ ,  $p$  is always satisfied until  $q$  is satisfied.
- $\text{EFp}$  describes the following property: There exists at least one path in  $M$ , such that  $p$  is eventually satisfied.

- AFp describes the following property: For each path in  $M$ ,  $p$  is eventually satisfied.
- EGp describes the following property: There exists at least one path in  $M$ , such that  $p$  is always satisfied.
- AGp describes the following property: For each path in  $M$ ,  $p$  is always satisfied.
- EXp describes the following property: There exists at least one path in  $M$ , such that  $p$  is satisfied in the next state.
- AXp describes the following property: For each path in  $M$ ,  $p$  is satisfied in the next state.

Given an arbitrary model  $M$ , how to use the DNA-computing-based method to determine whether the basic CTL constructs be satisfied by  $M$  or not? To this end, Sect. 3 will give such an approach (Fig. 1).



**Fig. 1.** An example on LFSA: systematic FSA model  $M_1$  which is used in experiments

## 2.2 Finite State Automata and Model Checking

**Definition 2.** A Finite State Automaton (FSA) is a five-tuples  $(\Sigma, Q, T, q_0, F)$ , where

- $\Sigma$  is a finite alphabet
- $Q$  is a finite set of states
- $T$  is a finite set of transitions:  $T : Q \times \Sigma \rightarrow R(Q)$
- $q_0 \in Q$  is an initial state
- $F \subseteq Q$  is a set of acceptance states

## 2.3 Sticker Automata and DNA Model Checking

### 2.3.1 Sticker Automata

As a model of DNA computing, a sticker automaton can realize a FSA. Given a DNA strand charactering an input string and a FSA, the sticker automaton can determine whether or not the string is accepted by the FSA.

#### 2.3.1.1 The Encoding Way of FSA and Input String

Reference [14] gives the following way of DNA encoding

Supposing  $M = (\Sigma, S, T, s_0, F)$  is a FSA, and every character  $a$  in the alphabet  $\Sigma$  can be encoded as  $C(a)$ , we have:

- (1) An input string  $a_1, \dots, a_n$  in  $\Sigma$  can be encoded with the following single-stranded DNA molecule:  $5' I_1 X_0 \dots X_m C(a_1) \dots X_0 \dots X_m C(a_n) X_0 \dots X_m I_2 3'$ , where  $I_1$  is an initiator sequence,  $X_0 \dots X_m$  is a spacer sequence separating  $C(a_i)$ , and  $I_2$  is a terminator sequence.
- (2) A transition  $T(s_i, a) = s_j$  is encoded as  $3' \overline{X_{i+1}} \dots \overline{X_m} \overline{C(a)} \overline{X_0} \dots \overline{X_j} 5'$ , where  $\overline{X}$  means the Watson-Crick complement (WC for short) of a nucleotide  $X$ ,  $\overline{C(a)}$  means the WC of the DNA strand charactering  $a$ .
- (3) An initial state  $s_i$  is encoded as  $3' \overline{I_1} \overline{X_0} \dots \overline{X_i} 5'$ .
- (4) An acceptance state  $s_j$  is encoded as  $3' \overline{X_{j+1}} \dots \overline{X_m} \overline{I_2} 5'$ .

### 2.3.1.2 The Process of DNA Computing Based on Sticker Automata

The computational process of sticker automata can be concluded as follows [14].

Step 1: Data preprocessing

- (1) Synthesize some DNA strands charactering an automaton and its input strings.
- (2) Put all the DNA strands into the test tube T, and anneal to make sure that the strands and their WC complements can be hybridized completely. The process of base pairing and the placement of ligase can form complete or partial double stranded DNA molecules.

Step 2: Computation

After the first step, we will see the following phenomena. If the input string is accepted by the automaton, the tube T contains only the complete double stranded DNA molecules which begin with an initiator sequence and terminate at a terminator sequence. Otherwise, there are partial double stranded DNA molecules or single stranded DNA molecules in T. Therefore, we add some ribozymes called Mung Bean into the test tube T, in order to degrade the single stranded DNA fragment, and retain the complete double stranded DNA molecules.

Step 3: Output of results

We can separate the different DNA molecules with different lengths using electrophoretic technique. If there exists a variety of length of DNA molecules, it indicates that there are some partial double stranded DNA molecules in T before we add the ribozymes, and the input string cannot be accepted by the automaton. Otherwise, T contains only complete double stranded DNA molecules before we add the ribozymes, and the input string can be accepted by the automaton.

### 2.3.2 DNA Model Checking

On the basis of sticker automata, Reference [13] presented a DNA-computing-based LTL model checking method, which can be denoted as the algorithm TL-MC-DNA(DNACODE(A),x), where DNACODE(A) and x are two input of the algorithm, A is a FSA expressing a run of a system, DNACODE(A) is an encoding with a sticker automaton for charactering A,  $x = \text{DNACODE}(A(f))$  is an encoding with a sticker automaton for charactering  $A(f)$ , and  $A(f)$  is a FSA model of a formula f. In Reference [13], the scope of f includes the all the basic LTL formulas and some popular LTL formulas, f formula for short. The output of the algorithm is yes or no, representing the result of the model checking.

### 3 The DNA Model Checking Method for the Basic CTL Constructs

If the encoding of sticker automaton which realizes a FSA of a system and the encoding of sticker automaton which realizes a FSA of a formula are inputted, the algorithm TL-MC-DNA(DNACODE(A), DNACODE(A(f))) in [13] can compute and return a result of model checking. This paper expands range of f, and a series of new encoding of sticker automata. By computing TL-MC-DNA(DNACODE(A), DNACODE(A(f''))), where  $f'' = \{\varphi_1\}$ , (f'' formula for short), we can perform DNA model checking for the one temporal logic formula. Ref. [13] has confirmed the effectiveness of the algorithm TL-MC-DNA for the f formulas by simulated biological experiments.

#### 3.1 The DNA Model Checking for the Four Universal Formulas

Comparing the CTL formula  $ApUq$  and the LTL formula  $pUq$ , we can clearly see that these two formulas have the same semantics. Therefore, we can use the algorithm TL-MC-DNA(DNACODE(A), x) to check the CTL formula  $ApUq$ . Similarly, the algorithm TL-MC-DNA(DNACODE(A), x) can also be employed to check the CTL constructs  $AFp$ ,  $AGp$  and  $AXp$ . The obtained algorithm is formulated as follows.

---

**Algorithm 1.** The DNA model checking algorithm for the universal CTL formulas CTLQ-MC-DNA(DNACODE(A), DNACODE(A( $f_q$ )))

**INPUT:** the encoding of sticker automaton which realizes a systematic FSA A, the encoding of sticker automaton which realizes a FSA of an universal CTL formula  $f_q$ , where  $f_q = ApUq, AFp, AGp$  or  $AXp$

**OUTPUT:** whether A satisfies  $f_q$ , or not

---

**BEGIN**

Step 1:

SELECT CASE  $f_q$

CASE  $ApUq$

g:= $pUq$  // where g is a f formula

CASE  $AFp$

g:= $Fp$  // where g is a f formula

CASE  $AGp$

g:= $Gp$  // where g is a f formula

CASE  $AXp$

g:= $Xp$  // where g is a f formula

ENDSELECT

Step 2:  $y := \text{TL-MC-DNA}(\text{DNACODE}(A), \text{DNACODE}(A(g)))$

Step 3: IF  $y = \text{"yes"}$ , THEN return "yes", ELSE return "no"

**END**

---

### 3.2 The DNA Model Checking for the Four Existence Formulas

The formula  $\text{EpUq}$  and the formula  $\text{ApUq}$  have the following relationship:  $\neg \text{EpUq} = \text{A} \neg p \bar{\bar{\text{U}}} \neg q$ . That is to say,  $\text{EpUq} = \neg \text{A} \neg p \bar{\bar{\text{U}}} \neg q$ . The formula  $\text{EGp}$  and the formula  $\text{AFp}$  have the following relationship:  $\neg \text{EGp} = \text{AF} \neg p$ , that is to say,  $\text{EGp} = \neg \text{AF} \neg p$ . The formula  $\text{EFp}$  and the formula  $\text{AGp}$  have the following relationship:  $\neg \text{EFp} = \text{AG} \neg p$ , that is to say,  $\text{EFp} = \neg \text{AG} \neg p$ . The formula  $\text{EXp}$  and the formula  $\text{AXp}$  have the following relationship:  $\neg \text{EXp} = \text{AX} \neg p$ , that is to say,  $\text{EXp} = \neg \text{AX} \neg p$ . Comparing  $\text{A} \neg p \bar{\bar{\text{U}}} \neg q$  and  $\varphi_1 = \neg p \bar{\bar{\text{U}}} \neg q$ , we can clearly see that these two formulas have the same semantics. Thus,  $\neg \varphi_1 = \text{EpUq}$ . Therefore, we can use the algorithm  $\text{TL-MC-DNA}(\text{DNACODE}(\text{A}))$  to check the CTL formula  $\text{EpUq}$ . Similarly, the algorithm  $\text{TL-MC-DNA}(\text{DNACODE}(\text{A}), x)$  can also be employed to check the CTL formulas  $\text{EFp}$ ,  $\text{EGp}$  and  $\text{EXp}$ . The obtained algorithm is formulated as follows.

---

**Algorithm 2.** The DNA model checking algorithm for the existence CTL formulas  $\text{CTL-MC-DNA}(\text{DNACODE}(\text{A}), \text{DNACODE}(\text{A}(f_c)))$

**INPUT:** the encoding of sticker automaton which realizes a systematic FSA  $\text{A}$ , the encoding of sticker automaton which realizes a FSA of an existence CTL formula  $f_c$ , where  $f_c = \text{EpUq}, \text{EFp}, \text{EGp}$  or  $\text{EXp}$

**OUTPUT:** whether  $\text{A}$  satisfies  $f_c$ , or not

---

**BEGIN**

Step 1:

SELECT CASE  $f_c$

CASE  $\text{EpUq}$

Step 1:  $g := \varphi_1$

Step 2:  $y := \text{TL-MC-DNA}(\text{DNACODE}(\text{A}), \text{DNACODE}(\text{A}(g)))$  //where  $g$  is a  $f''$  formula

Step 3: IF  $y = \text{"yes"}$ , THEN return "no", ELSE return "yes" //  $\varphi_1 = \neg(\text{EpUq})$

CASE  $\text{EFp}$

Step 1:  $g := G \neg p$

Step 2:  $y := \text{TL-MC-DNA}(\text{DNACODE}(\text{A}), \text{DNACODE}(\text{A}(g)))$  //where  $g$  is a  $f''$  formula

Step 3: IF  $y = \text{"yes"}$ , THEN return "no", ELSE return "yes" //  $G \neg p = \neg(\text{EFp})$

CASE  $\text{EGp}$

Step 1:  $g := F \neg p$

Step 2:  $y := \text{TL-MC-DNA}(\text{DNACODE}(\text{A}), \text{DNACODE}(\text{A}(g)))$  //where  $g$  is a  $f''$  formula

Step 3: IF  $y = \text{"yes"}$ , THEN return "no", ELSE return "yes" //  $F \neg p = \neg(\text{EGp})$

CASE  $\text{EXp}$

Step 1:  $g := X \neg p$

Step 2:  $y := \text{TL-MC-DNA}(\text{DNACODE}(\text{A}), \text{DNACODE}(\text{A}(g)))$  //where  $g$  is a  $f''$  formula

Step 3: IF  $y = \text{"yes"}$ , THEN return "no", ELSE return "yes" //  $X \neg p = \neg(\text{EXp})$

ENDSELECT

**END**

---

### 3.3 The DNA Model Checking for the Basic CTL Constructs

The principle of this algorithm is: (1) If a basic CTL construct is an universal formula, the Algorithm 1 will be called. (2) And if a basic CTL construct is an

existence formula, the Algorithm 2 will be called. In this way, the model checking of the basic CTL constructs can be conducted. The algorithm is formulated as follows.

---

**Algorithm 3.** The DNA model checking algorithm for the basic CTL constructs CTL-MC-DNA(DNACODE(A), DNACODE(A( $f_{CTL}$ )))

**INPUT:** the encoding of sticker automaton which realizes a systematic FSA A, the encoding of sticker automaton which realizes a FSA of a basic CTL construct  $f_{CTL}$

**OUTPUT:** whether A satisfies  $f_{CTL}$ , or not

---

**BEGIN**

Step 1: IF there exists  $f_c$ , such that  $f_{CTL}=f_c$ ,

THEN call CTLC-MC-DNA(DNACODE(A), DNACODE(A( $f_c$ )))

ELSEIF there exists  $f_q$ , such that  $f_{CTL}=f_q$ ,

THEN call CTLQ-MC-DNA(DNACODE(A), DNACODE(A( $f_q$ )))

ENDSELECT

**END**

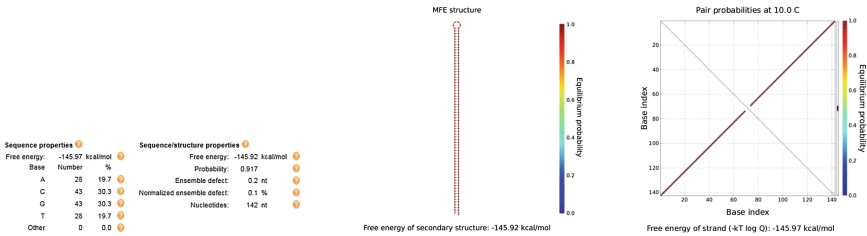
---

## 4 Simulated Experiments

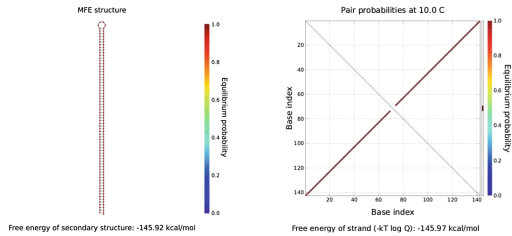
**Experimental platform:** NUPACK [15]

**Experimental procedure:** (1) one can design the encoding of the sticker automata for the systematic FSA, as well as the encoding of the sticker automata for the FSA of formula, respectively; (2) for these FSAs mentioned above, one can simulate the process of hybridization between some single stranded DNA molecules; (3) according to the algorithms proposed in this paper, one can get the results of model checking of the several formulas, by reading the results of hybridization.

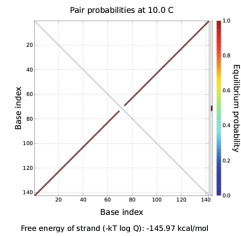
**Experimental objective:** To test the correctness, effectiveness and biological realizability of the new algorithms.



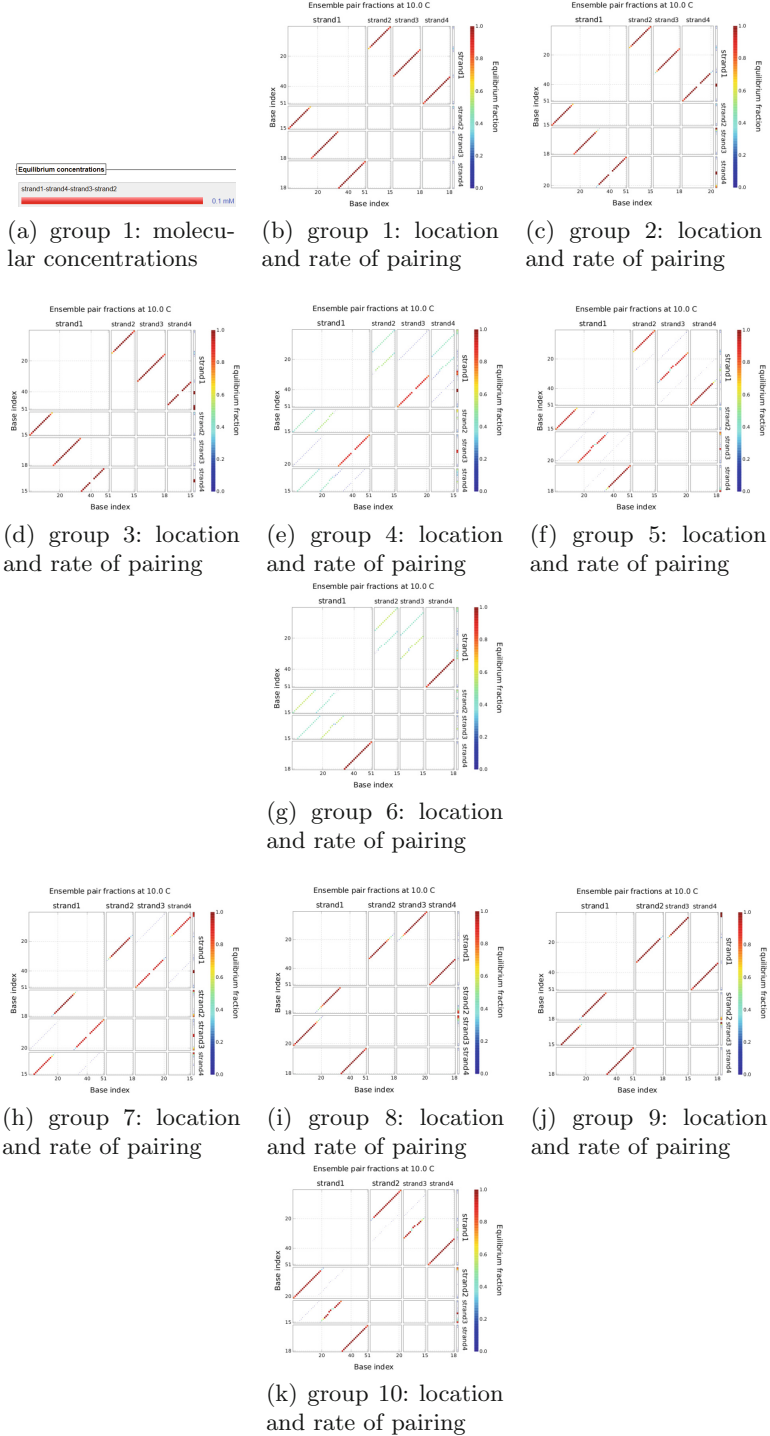
**Fig. 2.** checking the formula  $\varphi_1$ : the structural properties of encoding sequence



**Fig. 3.** (Chromatic) thermodynamic analysis for  $\varphi_1$ : minimum free energy structure



**Fig. 4.** (Chromatic) checking for  $\varphi_1$ : pairing probability in equilibrium



**Fig. 5.** (Chromatic) checking for  $\varphi_1$ : the groups of sub-experimental results on base pairing and hybridizations



We have conducted the simulated experiments for  $\varphi_1$ . We have designed a DNA encoding via NUPAC. Figures 2, 3 and 4 show the thermodynamic analysis of the encoding sequence at 10 Celsius degree. We have checked whether or not the systematic FSA  $M_1$  satisfies the formula  $\varphi_1$ . The results are shown in Fig. 5.

## 5 Conclusions

Early researches on DNA computing focus on the models and algorithms based on non autonomous. In recent years, the DNA computing technique has developed to the self-assembly. The main results of this paper are the Algorithm 3, which is based on the self assembly of sticker automata. With these algorithms at hands, we can conduct model checking for the basic CTL constructs via some DNA molecules. This is the main contribution of this paper.

**Acknowledgment.** This work has been supported by the NSFC project under Grant U1204608, 61572444, 61272022 and 61472372.

## References

1. Clarke, E., et al.: Model Checking. MIT press, Cambridge (1999)
2. Pnueli, A.: The temporal logic of programs. In: Symposium on Foundations of Computer Science, Washington, DC, USA, pp. 46–57 (1977)
3. Benari, M., Pnueli, A., Manna, Z.: The temporal logic of branching time. *Acta Informatica* **20**(3), 207–226 (1983)
4. Emerson, E., Clarke, E.: Using branching time temporal logic to synthesize synchronization skeletons. *Sci. Comput. Program.* **2**(3), 241–266 (1982)
5. Adleman, L.: Molecular computation of solutions to combinatorial problems. *Science* **266**(5187), 1021–1023 (1994)
6. Lipton, R.: DNA solution of hard computational problems. *Science* **268**(5210), 542–545 (1995)
7. Ouyang, Q., Kaplan, P.D., Liu, S., et al.: DNA solution of the maximal clique problem. *Science* **278**(17), 446–449 (1997)
8. Shapiro, E., Benenson, Y., Adar, R., et al.: Programmable and autonomous computing machine made of biomolecules. *Nature* **414**(6862), 430–434 (2001)
9. Jin, X.: Probe machine. *IEEE Trans. Neural Netw. Learn. Syst.* **27**(7), 1405–1416 (2016)
10. Jin, X.: Forthcoming era of biological computer. *Bull. Chin. Acad. Sci.* **29**(1), 42–54 (2014). (in Chinese)
11. Emerson, E., Hager, K., Konieczka, J.: Molecular model checking. *Int. J. Found. Comput. Sci.* **17**(04), 733–741 (2006)
12. Zhu, W.-J., Zhou, Q.-L., Li, Y.-L.: LTL model checking based on DNA computing. *Acta Electronica Sinica* **44**(6), 1265–1271 (2016)
13. Zhu, W.-J., Zhou, Q.-L., Zhang, Q.-X.: A LTL model checking approach based on DNA computing. *Chin. J. Comput.* (2016)
14. Zimmermann, K., Ignatova, Z., Perez, M.: *DNA Computing Models*. Springer, New York (2008)
15. NUPACK (2015). <http://www.nupack.org/partition/new>

Bio-inspired Computing – Theories and Applications  
11th International Conference, BIC-TA 2016, Xi'an,  
China, October 28-30, 2016, Revised Selected Papers,  
Part I

Gong, M.; Pan, L.; Song, T.; Zhang, G. (Eds.)

2016, XX, 528 p. 189 illus., Softcover

ISBN: 978-981-10-3610-1