Hindawi
BioMed Research International
Volume 2017, Article ID 7941845, 33 pages
https://doi.org/10.1155/2017/7941845

Research Article

Model Checking Temporal Logic Formulas Using Sticker Automata

Weijun Zhu,[1] Changwei Feng,[2] and Huanmei Wu[3]

[1]*School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China*
[2]*The Second Affiliated Hospital of Zhengzhou University, Zhengzhou 450001, China*
[3]*School of Informatics and Computing, Indiana University-Purdue University Indianapolis, Indianapolis, IN, USA*

Correspondence should be addressed to Weijun Zhu; zhuweijun76@163.com

Received 4 October 2016; Revised 13 February 2017; Accepted 18 April 2017; Published 28 September 2017

Academic Editor: J. R. Torregrosa

As an important complex problem, the temporal logic model checking problem is still far from being fully resolved under the circumstance of DNA computing, especially Computation Tree Logic (CTL), Interval Temporal Logic (ITL), and Projection Temporal Logic (PTL), because there is still a lack of approaches for DNA model checking. To address this challenge, a model checking method is proposed for checking the basic formulas in the above three temporal logic types with DNA molecules. First, one-type single-stranded DNA molecules are employed to encode the Finite State Automaton (FSA) model of the given basic formula so that a sticker automaton is obtained. On the other hand, other single-stranded DNA molecules are employed to encode the given system model so that the input strings of the sticker automaton are obtained. Next, a series of biochemical reactions are conducted between the above two types of single-stranded DNA molecules. It can then be decided whether the system satisfies the formula or not. As a result, we have developed a DNA-based approach for checking all the basic formulas of CTL, ITL, and PTL. The simulated results demonstrate the effectiveness of the new method.

1. Introduction

Differing from an electronic computer, a DNA computer uses DNA molecules as the carrier of computation. In 1994, a Turing Award winner Professor Adleman published an article in ⟨Science⟩ that solved a small-scale Hamiltonian path problem with a DNA experiment [1], which is regarded as the pioneering work in the field of DNA computing. As DNA computing has a huge advantage for parallel processing, this technique was subsequently advanced rapidly. Many models and approaches based on DNA computing have been developed to solve some complex computational problems, especially the famous NP-hard problems and PSPACE-hard ones. For example, Lipton published an article in ⟨Science⟩ that improved Adleman's idea for the SAT problem [2]. Ouyang et al. published an article in ⟨Science⟩ that presented a DNA-computing-based model for solving the maximal clique problem [3]. Benenson et al. published an article in ⟨Nature⟩ that solved an automata problem of two states and two characters using the autonomous DNA computing technique [4].

Many other DNA models have been constructed, such as the restricted model [5], the sticker system [6], the length-encoding model [7], the sticker automaton model [8], the DNA Turing machine model [9], the nonenumerative DNA model [10], the giant-magneto-resistance-based DNA model [11], the logical DNA molecular model [12], and the logical nanomolecular model [13]. And a series of methods based on nonautonomous or self-assembling are proposed for solving various complex computational problems, including the Nondeterministic Polynomial (NP) ones. For example, there are methods proposed for the maximum clique problem [14, 15], the vertex coloring one [10, 16], the SAT one [11], the N queen one [17], the maximum matching one [18], the minimum vertex cover one [19], the minimum and exact cover one [20], the subset-sum one [21], the classical Ramsey number one [22], the spatial cluster analysis [23, 24], and the knapsack [25].

On the one hand, some problems in computer science can be solved by applying the techniques based on biochemical reactions in test tubes, nanodevices, or molecular self-assembly [1, 26–28]. On the other hand, due to the

Table 1: State of the art of DNA model checking and its open problems.

Temporal logic	State of the art (the formulas which have been checked)	The problems to be solved (the formulas which cannot be checked yet)
LTL	All the four basic formulas	General formulas and cellular model checking
CTL	The basic formula EFp	Another seven basic formulas (will be studied in this paper)
ITL	Nothing reported	All the two basic formulas (will be studied in this paper)
PTL	Nothing reported	All the one formula (will be studied in this paper)
DC	It is infeasible due to the limitation of the current biochemical experimental technique	

excellent information processing mechanism and the huge parallelism, some living cells can also be employed to perform some computations. The site-specific DNA recombinase Hin, which can mediate inversion of DNA segments that represent variables, was used to produce the solution. In this model, each cell can produce and examine a solution of satisfiability problem. As a result, billions of cells can explore billions of possible solutions [29]. In this way, Professor Chen et al. constructed a cellular computing model [29] to solve the satisfiability problem. In addition, a conditional learning system in *Escherichia coli* was built to identify the "bad man" signal with the help of the "learning" signal. It is a useful attempt to construct the artificial intelligent system using some molecular biological techniques.

One of the key differences between computer and other computing tools is the universality. Professor Xu constructed a mathematical model called "probe machine" for the general DNA computer [30]. By integrating the storage system, operation system, detection system, and control system into a whole, a real general DNA computer was gradually obtained, which was the "Zhongzhou DNA computer" [30]. 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 [15]. It is a universal DNA computing model which can be realized in biology. And a Turing machine is just a "special case" of a probe machine [15]. This significant progress has raised the practical importance of the researches on DNA computing.

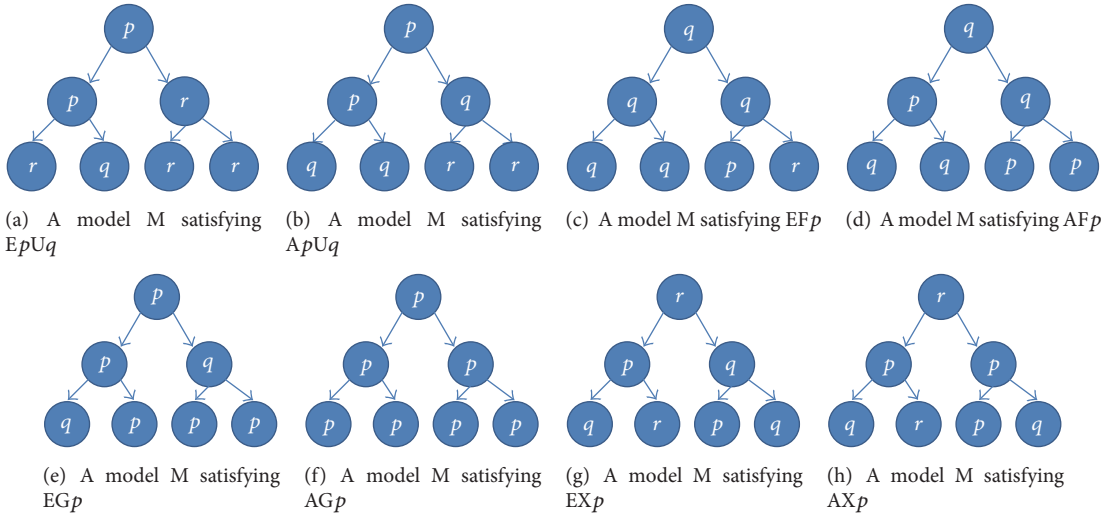
More studies on DNA computing have been conducted for the last three years. Some of the major studies are summarized as follows: (1) aiming to deal with some inherent flaws of DNA computing, such as adaptability [31] and instability [32]; (2) employing DNAs to realize some basic computing components and/or techniques, such as data storage [33], database operations [34], odd parity checker [35], half adder [36], encryption [37], and data hiding [38]; (3) utilizing DNAs to address some problems in real world, such as the inverse kinematics redundancy problem of six-degree-of-freedom humanoid robot arms [39], dynamic control of elevator systems [40], and hyperspectral remote sensing data/imagery [41, 42].

Besides the satisfiability problem, model checking (MC) is another important computational problem. These two problems are correlated. The MC proposed by the Turing Award winner Professor Clarke et al. [43] is widely used in the fields of CPU verification [44], network protocol verification, security protocol verification [45], and software verification [46]. MC algorithms answer automatically the question of whether a system satisfies the given property or not. NASA, Intel, IBM, and Motorola are using this technique. The general principles of MC can be given as follows: (i) a system model is constructed with an automaton; (ii) a property which the system should satisfy is described by a temporal logic formula; and (iii) if an automaton is a model of the formula, the system model satisfies the property; otherwise, the system does not satisfy the property.

In order to describe the different temporal properties, some different temporal logic types have been proposed. For instance, Linear Temporal Logic (LTL) was introduced into computer science to express the linear properties by the Turing Award winner Professor Pnueli [47]. Computation Tree Logic (CTL) was proposed to express the branch properties by the Turing Award winner Professor Clarke [48, 49]. Interval Temporal Logic (ITL), Duration Calculus (DC), and Projection Temporal Logic (PTL) were also investigated to express other temporal properties [50–52].

As a complex computational problem, model checking under the circumstance of DNA computing is always a goal for researchers. In 2006, some DNA molecules were applied to conduct CTL model checking for the first time by the Turing Award winner Professor Emerson et al. [53]. However, this method can check only one basic CTL formula, called EFp. It is known that there are eight basic formulas in CTL, that is, EpUq, ApUq, EFp, AFp, EGp, AGp, EXp, and AXp. It has been a pending and challenging issue to perform model checking for all of the eight basic CTL formulas using DNA computing. As shown in Table 1, there are eight basic formulas for CTL, two for ITL, and one for PTL. Except for the EFp formula, all the other ten basic formulas in CTL, ITL, and PTL cannot conduct model checking under the circumstance of DNA computing using the existing methods.

Motivated by it, we proposed a set of DNA-based model checking algorithms. With our new algorithms, all the eleven basic formulas for CTL, ITL, and PTL can undergo model checking via some DNA molecules. Basically, the core model checking problem for the CTL, ITL, and PTL is solved by DNA computing, because every CTL/ITL/PTL formula can be obtained by combining the basic CTL/ITL/PTL formulas recursively. This is the main contribution of this paper.



(a) A model M satisfying EpUq (b) A model M satisfying ApUq (c) A model M satisfying EFp (d) A model M satisfying AFp

(e) A model M satisfying EGp (f) A model M satisfying AGp (g) A model M satisfying EXp (h) A model M satisfying AXp

Figure 1: Examples of the basic CTL formulas and their models.

The rest of this paper is organized as follows. Section 2 introduces some basic concepts. Our newly proposed algorithms will be described in Section 3. The simulated experiments will be presented in Section 4, which demonstrates that the new algorithms are feasible in molecular biology. Section 5 provides brief conclusions. The formal definitions of these temporal logic types are given in the Appendix.

2. Preliminary

2.1. The Basic Formulas in CTL [43]

Definition 1. Let p and q be atomic propositions and EpUq, ApUq, EFp, AFp, EGp, AGp, EXp, and AXp be the basic CTL formulas. An arbitrary CTL formula can be obtained by recursive combinations of these basic CTL formulas.

An atomic proposition and a basic CTL formula are interpreted on a system model M, and their intuitive meanings are given as follows:

(i) p or q is satisfied in a state s.

(ii) EpUq describes the property: there exists at least one path in M, such that p is always satisfied until q is satisfied.

(iii) ApUq describes the property: for each path in M, p is always satisfied until q is satisfied.

(iv) EFp describes the property: there exists at least one path in M, such that p is eventually satisfied.

(v) AFp describes the property: for each path in M, p is eventually satisfied.

(vi) EGp describes the property: there exists at least one path in M, such that p is always satisfied.

(vii) AGp describes the property: for each path in M, p is always satisfied.

(viii) EXp describes the property: there exists at least one path in M, such that p is satisfied in the next state.

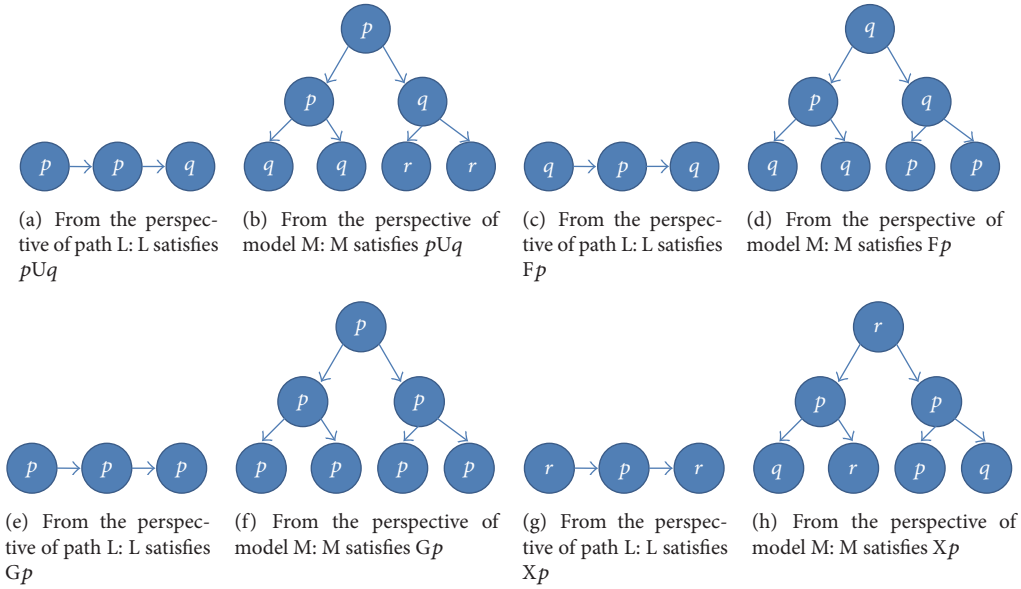
(ix) AXp describes the property: for each path in M, p is satisfied in the next state.

Figure 1 gives some example models which satisfy the eight basic CTL formulas. A circle represents a state, and a letter in a circle represents an atomic proposition which is satisfied in the state. A line segment with an arrow means an edge (i.e., a transition between two states). A state sequence from the root node to a leaf node is called a path. Time passes from top to bottom, and the different branches represent the alternative transitions from the current state to the next one.

For the model M in Figure 1(a), there are four paths. Each path passes through three states at three moments, which forms four sequences of atomic propositions: ppr, ppq, prr, and prr. It is noticeable that ppq, that is, the second path, satisfies the following property: p is always satisfied until q is satisfied. In contrast, any other path in M does not satisfy this property. According to the definition of EpUq, the model M satisfies EpUq.

For the model M in Figure 1(b), there are also four paths with three states for each path. The four sequences of atomic propositions are: ppq, ppq, pqr and pqr. All paths in M satisfy the property: p is always satisfied until q is satisfied. According to the definition of ApUq, the model M satisfies ApUq.

Similarly, for the model M in Figure 1(c), the path qqp in M satisfies the property: p is eventually satisfied. Thus, the model M satisfies EFp. For the model M in Figure 1(d), all four paths, qpq, qpq, qqp, and qqp, satisfy the property: p is eventually satisfied. Thus, the model satisfies AFp. For the model M in Figure 1(e), the path ppp in M satisfies the property: p is always satisfied. Thus, the model M satisfies EGp. For the model M in Figure 1(f), all the four paths, ppp, ppp, ppp, and ppp, satisfy the property: p is always satisfied, which makes the model M satisfy AGp. For the model M in Figure 1(g), the paths rpq and rpr in M satisfy the property: p is satisfied in the next state. Thus, M satisfies EXp. For the model M in Figure 1(h), all the four paths, rpq, rpr, rpp, and



(a) From the perspective of path L: L satisfies pUq

(b) From the perspective of model M: M satisfies pUq

(c) From the perspective of path L: L satisfies Fp

(d) From the perspective of model M: M satisfies Fp

(e) From the perspective of path L: L satisfies Gp

(f) From the perspective of model M: M satisfies Gp

(g) From the perspective of path L: L satisfies Xp

(h) From the perspective of model M: M satisfies Xp

Figure 2: Examples of the basic LTL formulas and their models.

rpq, satisfy the property: p is satisfied in the next state. Thus, this M satisfies AXp.

Given an arbitrary model M, the challenge is how to use the DNA-computing-based method to determine whether the eight basic CTL formulas are satisfied by M or not. Section 3.1 will provide our new approach which can check all the eight basic CTL formulas.

2.2. The Basic Formulas in LTL [43]

Definition 2. Let p and q be atomic propositions and pUq, Fp, Gp, and Xp be the basic LTL formulas. An arbitrary LTL formula can be obtained by combining recursively some basic LTL formulas. An atomic proposition and a basic LTL formula are interpreted on a path L and a system model M, and their intuitive meanings are given as follows:

(i) p or q is satisfied in a state s, or not.

(ii) pUq describes the property: for each path L in M, p is always satisfied until q is satisfied.

(iii) Fp describes the property: for each path L in M, p is eventually satisfied.

(iv) Gp describes the property: for each path L in M, p is always satisfied.

(v) Xp describes the property: for each path L in M, p is satisfied in the next state.

For a path L, time passes from left to right, and the system transits from the current state to the next one. For a model M, time passes from top to bottom, and the different branches represent the alternative transitions from the current state to the next one.

Figure 2 gives one sample M for each basic LTL formula, respectively. The formula pUq is called the core LTL formula since every basic LTL formula can be expressed by pUq. Given an arbitrary model M, previous studies have provided approaches on how to use the DNA-computing-based method to determine whether the four basic LTL formulas are satisfied by M or not [54, 55].

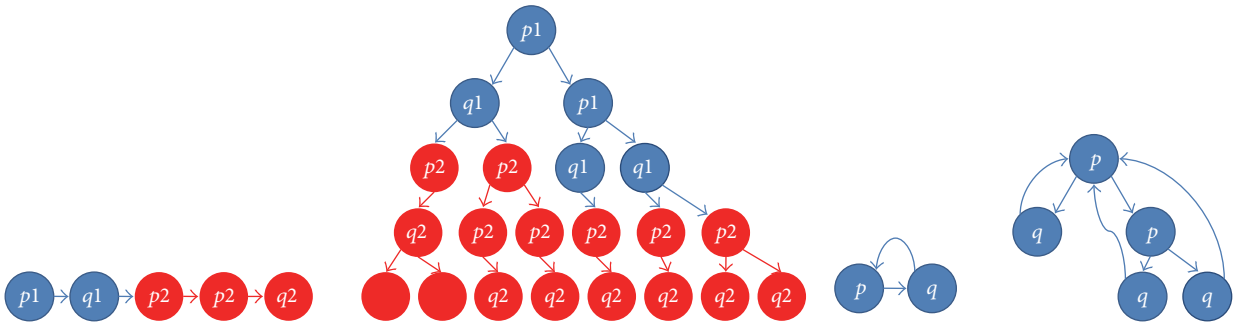
2.3. The Basic Formulas in ITL [50]

Definition 3. Let p, q, p_1, p_2, q_1, and q_2 be atomic propositions and $(p_1Uq_1);(p_2Uq_2)$ and $(pUq)^*$ be the basic ITL formulas. An arbitrary ITL formula can be obtained by combining recursively some basic ITL formulas. A basic ITL formula is interpreted on a path L and a system model M, and their intuitive meanings are given as follows.

(i) $(p_1Uq_1);(p_2Uq_2)$: for each path L in M, the following property holds: prefix subpath (i.e., prefix interval) satisfies the core LTL formula p_1Uq_1, and suffix subpath (i.e., suffix interval) satisfies the core LTL formula p_2Uq_2.

(ii) $(pUq)^*$: for each path L in M, the following property holds: L circulates in a loop body consisting of a subpath (i.e., a loop body consisting of an interval), and the loop body of interval satisfies the core LTL formula pUq.

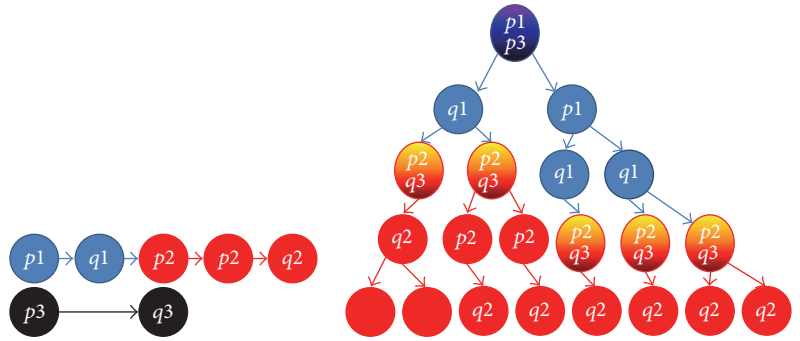
Figure 3 gives an example for each basic ITL formula, respectively. For $(p_1Uq_1);(p_2Uq_2)$, any path L in M has the following characteristics: L reaches a number of red states after it crosses some blue states. The prefix interval of L is denoted as the state sequence marked in blue, whereas the suffix interval of L is denoted as the state sequence marked in red. An ITL formula is satisfied in such an interval. In Figure 3(a), $p_1q_1p_2p_2q_2$ is a path satisfying the ITL formula $(p_1Uq_1);(p_2Uq_2)$. The prefix interval of $p_1q_1p_2p_2q_2$ satisfies the LTL formula p_1Uq_1, whereas the suffix interval of $p_1q_1p_2p_2q_2$ satisfies the LTL formula p_2Uq_2. Similarly,



(a) From the perspective of path L and model M: they satisfy $(p_1 \mathrm{U} q_1); (p_2 \mathrm{U} q_2)$

(b) From the perspective of path L and model M: they satisfy $(p \mathrm{U} q)^*$

FIGURE 3: Examples of the basic ITL formulas and their models.



(a) From the perspective of path L

(b) From the perspective of model M

FIGURE 4: An example of the basic PTL formula $((p_1 \mathrm{U} q_1), (p_2 \mathrm{U} q_2))$ prj $(p_3 \wedge \mathrm{X} q_3)$ and its model.

all the paths in M of Figure 3(a) satisfy the ITL formula $(p_1 \mathrm{U} q_1); (p_2 \mathrm{U} q_2)$. In Figure 3(b), $pqpq \ldots$ is a path satisfying the ITL formula $(p \mathrm{U} q)^*$. The loop body consisting of an interval (i.e., pq) satisfies the LTL formula $p \mathrm{U} q$. Similarly, all the paths in M of Figure 3(b) satisfy the ITL formula $(p \mathrm{U} q)^*$.

Given an arbitrary model M, we will provide a new solution in Section 3.2 on how to use the DNA-computing-based method to determine whether the two basic ITL formulas are satisfied by M or not.

2.4. The Basic Formula in PTL [52]

Definition 4. Let p_1, p_2, p_3, q_1, q_2, and q_3 be atomic propositions and $((p_1 \mathrm{U} q_1), (p_2 \mathrm{U} q_2))$ prj $(p_3 \wedge \mathrm{X} q_3)$ be the basic PTL formula. An arbitrary PTL formula can be obtained by combining recursively the basic PTL formula. A basic PTL formula is interpreted on a path L and a system model M, and its intuitive meaning is given as follows:

(i) $((p_1 \mathrm{U} q_1), (p_2 \mathrm{U} q_2))$ prj $(p_3 \wedge \mathrm{X} q_3)$: for each path L in M, the following property holds: (1) the prefix subpath (i.e., the fined-grained prefix interval) satisfies the core LTL formula $p_1 \mathrm{U} q_1$, (2) the suffix subpath (i.e., the fined-grained suffix interval) satisfies the core LTL formula $p_2 \mathrm{U} q_2$, and (3) the state sequence consisting of the first state in the fine-grained prefix interval and the first state in the fine-grained suffix interval (i.e., the coarse-grained interval) satisfies the LTL formula $p_3 \wedge \mathrm{X} q_3$.

Figure 4 gives a sample model for the basic PTL formula. For $((p_1 \mathrm{U} q_1), (p_2 \mathrm{U} q_2))$ prj $(p_3 \wedge \mathrm{X} q_3)$, any path L in M has the three intervals: the fined-grained prefix interval is made up of the blue states, the fined-grained suffix interval is made up of the red states, and the coarse-grained interval is made up of the black states. In fact, the difference of the fined-grained intervals and the coarse-grained interval is the different units of time elapse.

Given an arbitrary model M, we will provide a new solution in Section 3.3 on how to use the DNA-computing-based method to determine whether the basic PTL formula is satisfied by M or not.

2.5. Finite State Automata and Model Checking

Definition 5. A Finite State Automaton (FSA) is a five-tuples (Σ, Q, T, q_0, F), where

(i) Σ is a finite alphabet,

(ii) Q is a finite set of states,

(iii) T is a finite set of transitions: $T : Q \times \Sigma \to R(Q)$,

(iv) $q_0 \in Q$ is an initial state,

(v) $F \subseteq Q$ is a set of acceptance states.

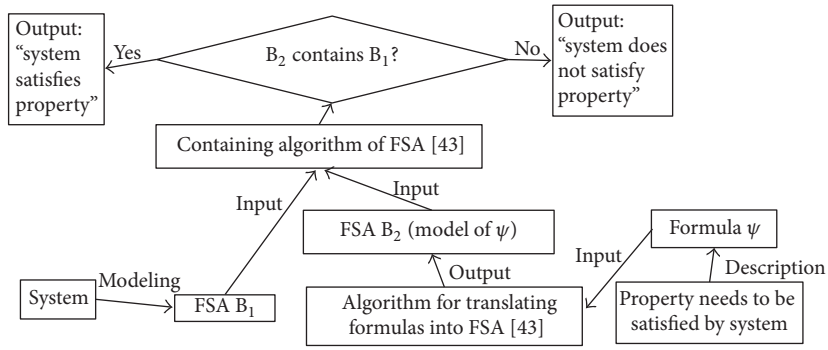


Figure 5: Principle of the model checking algorithms based on classic computing.

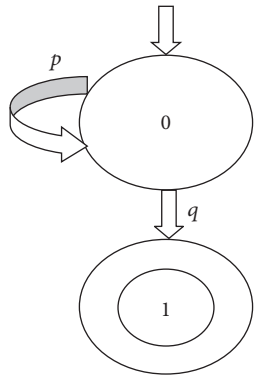


Figure 6: An example on FSA.

Figure 6 depicts an example for an FSA. This automaton is made up of two states and two transitions. State 0 is an initial state which is pointed at by an arrow without source, whereas state 1 is an acceptance state which is marked by a double circle. The automaton will enter state 0 if p is input at state 0, whereas the automaton will enter state 1 if q is input at state 0. The string pq is an acceptance word, since the automaton will transit from an initial state to an acceptance state if pq is input. Similarly, the strings q, ppq, $pppq$, ... are acceptance words too. An acceptance language of an automaton is made up of all of the acceptance words of the automaton. In this example, $\{q, pq, ppq, pppq, \ldots\}$ is the acceptance language of the automaton which is illustrated by Figure 6.

The only difference between the automaton in Figure 6 and the one in Figure 7 is that the atomic propositions in the latter automaton are satisfied in the states rather than in the transitions. Therefore, the latter automaton is called a Label FSA (LFSA).

In classical computation, the principles of the algorithms for temporal logic model checking can be illustrated by Figure 5. A LFSA, denoted as B_1, is used to describe some behaviors of a system, whereas an FSA, denoted as B_2, is employed to construct a model of a temporal logic formula. The model checking algorithm will decide that the system meets the property specified by the formula, if some inclusion relations hold between the two acceptance languages of the two automata.

2.6. Sticker Automata and DNA Model Checking

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

$M = (\Sigma, S, T, s_0, F)$ is an FSA, and every character a in the alphabet Σ can be encoded as $C(a)$. One way of the DNA encoding is as follows [56]:

(1) An input string $a_1, \ldots, a_n$ in Σ can be encoded with the single-stranded DNA molecule: $5' I_1 X_0 \cdots X_m C(a_1) \cdots X_0 \cdots X_m C(a_n) X_0 \cdots X_m I_2 3'$, where I_1 is an initiator sequence, $X_0 \cdots X_m$ 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 $3' \overline{X_{i+1} \cdots X_m}\, \overline{C(a)}\, \overline{X_0 \cdots X_j}\, 5'$, where $\overline{X}$ means the Watson-Crick complement (WC for short) of a nucleotide X and $\overline{C(a)}$ means the WC of the DNA strand characterizing a.

(3) An initial state s_i is encoded as $3'\ \overline{I_1}\, \overline{X_0 \cdots X_i}\, 5'$.

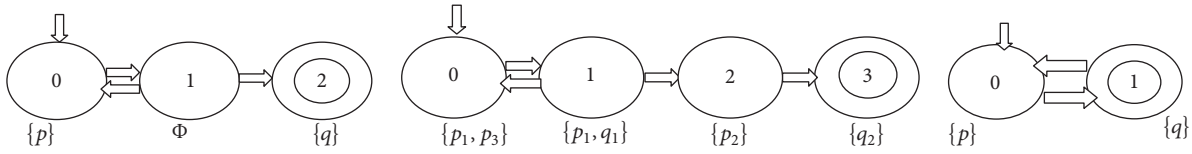
(4) An acceptance state s_j is encoded as $3'\ \overline{X_{j+1} \cdots X_m}\, \overline{I_2}\, 5'$.

The computational process of sticker automata can be summarized in the following three steps [56].

Step 1 (data preprocessing). (1) Synthesize some DNA strands characterizing 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 Step 1, there are two possible cases. 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 or single-stranded DNA molecules in T. For the second case, some fragments of the single-stranded DNA molecules which characterize the input strings



(a) A systematic FSA M_1 (b) A systematic FSA M_2 (c) A systematic FSA M_3

Figure 7: Some examples on LFSA: the systematic models of the experiments in this paper.

Table 2: Relationships: the four formulas of temporal logic and their FSA models, where $\bar{\mathrm{U}}$ is the logical duality of U.

The formula	$\varphi_1 = \neg p\ \bar{\mathrm{U}}\ \neg q$	$\varphi_2 = (p_1 \mathrm{U} q_1); (p_2 \mathrm{U} q_2)$	$\varphi_3 = (p \mathrm{U} q)^*$	$\varphi_4 = ((p_1 \mathrm{U} q_1), (p_2 \mathrm{U} q_2))\ \mathrm{prj}\ (p_3 \wedge \mathrm{X} q_3)$
FSA of formula	A_1	A_2	A_3	A_4

are paired successfully with some single-stranded DNA molecules which characterize transitions, whereas other fragments of the single-stranded DNA molecules which characterize the input strings cannot be paired with any single-stranded DNA molecules which characterize transitions. Therefore, ribozymes called Mung Bean are poured into the test tube T to degrade the single-stranded DNA fragment and retain the complete double-stranded DNA molecules.

Step 3 (output of results). The DNA molecules with different lengths can be separated using the electrophoretic technique. If there exist a variety of lengths of DNA molecules, this 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.

2.6.2. DNA Model Checking. On the basis of sticker automata, a DNA-computing-based LTL model checking method has been presented [55], which can be denoted as algorithm TL-MC-DNA(DNACODE(A), x), where DNACODE(A) and x are two inputs of the algorithm, where A is an FSA expressing a run of a system, DNACODE(A) is an encoding with a sticker automaton for characterizing A, x = DNACODE(A(f)) is an encoding with a sticker automaton for characterizing A(f), and A(f) is an FSA model of a formula f. The scope of f includes all the basic LTL formulas and some popular LTL formulas (f formula) [55]. The output of the algorithm is yes or no, representing the result of the model checking. The principle of this algorithm is illustrated by Figure 8.

2.7. The Four FSAs of the Formulas and Their DNA Model Checking. Given a temporal logic formula, an FSA model can be computed [43, 50, 52]. Figure 9 gives the four FSA models for the four specific formulas of temporal logic, respectively. Their corresponding relations are shown in Table 2, where φ_2 and φ_3 are the basic ITL formulas and φ_4 is the basic PTL formula. In addition, $\neg$ is logical negation, $\bar{\mathrm{U}}$ is logical duality of U, and $\neg p\ \bar{\mathrm{U}}\ \neg q$ describes the property: for each path in M, there exists at least one state which does not satisfy p, before q is satisfied.

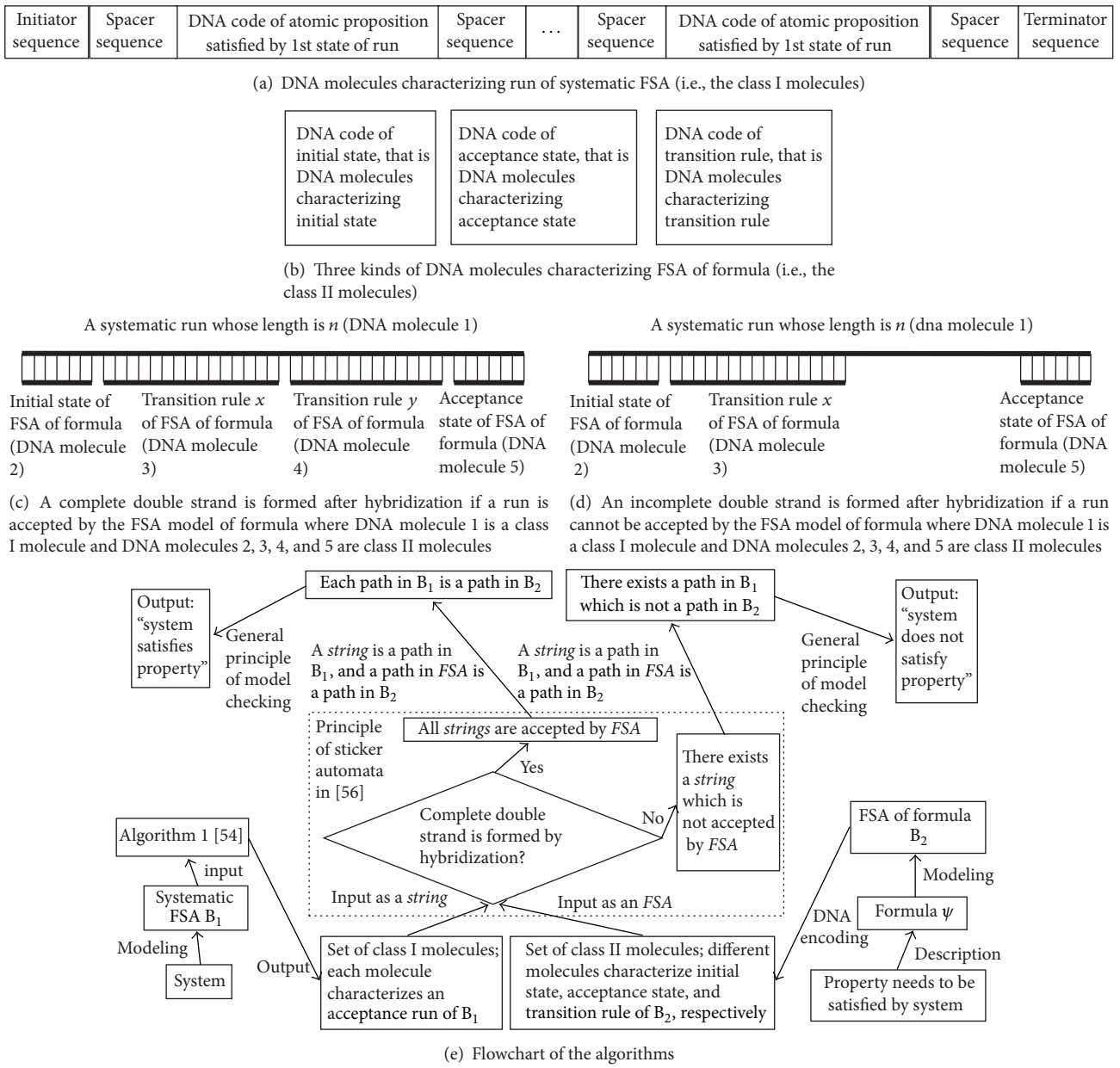
3. The DNA Model Checking Method

As mentioned in Section 2.6.2, if the encoding of one sticker automaton for an FSA of a system and the encoding of the other sticker automaton for an FSA of a formula are input into the algorithm TL-MC-DNA(DNACODE(A), DNACODE(A(f))) [55], the algorithm can compute and return the model checking results. This has been confirmed for the effectiveness of the algorithm TL-MC-DNA for the f formulas by simulated biological experiments [55]. This paper expands the range of the formula f and a series of new encodings of sticker automata, which will be explained in detail in Section 4. The DNA model checking for the four temporal logic formulas in Table 2 is performed by running the algorithm TL-MC-DNA(DNACODE(A), DNACODE(A(f''))), where $f'' = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$ (f'' formula). Section 4 will study the effectiveness of the new algorithm for the f'' formulas by a number of simulated biological experiments. It should be noted that the algorithm TL-MC-DNA(DNACODE(A), DNACODE(A(f))) comes from previous research [55]. Due to space limitations, its pseudocode is not given in this paper.

3.1. The DNA Model Checking for the Basic CTL Formulas. There are eight basic CTL formulas. Due to the different semantics, the DNA model checking algorithms are different too.

3.1.1. The DNA Model Checking for the Four Universal Formulas. Four basic CTL formulas, ApUq, AFp, AGp, and AXp, are called the universal formulas since their semantics are all involved in "all paths." Comparing the CTL formula ApUq and the LTL formula pUq, it can be clearly seen that these two formulas have the same semantics. Therefore, the algorithm TL-MC-DNA(DNACODE(A), x) [55] can be employed to check the CTL formulas ApUq, AFp, AGp, and AXp. The detailed algorithm is formulated as shown in Algorithm 1.

3.1.2. The DNA Model Checking for the Four Existence Formulas. The remaining four basic CTL formulas, EpUq, EFp,



(a) DNA molecules characterizing run of systematic FSA (i.e., the class I molecules)

(b) Three kinds of DNA molecules characterizing FSA of formula (i.e., the class II molecules)

(c) A complete double strand is formed after hybridization if a run is accepted by the FSA model of formula where DNA molecule 1 is a class I molecule and DNA molecules 2, 3, 4, and 5 are class II molecules

(d) An incomplete double strand is formed after hybridization if a run cannot be accepted by the FSA model of formula where DNA molecule 1 is a class I molecule and DNA molecules 2, 3, 4, and 5 are class II molecules

(e) Flowchart of the algorithms

Figure 8: Principle of the model checking algorithms based on DNA computing [55].

EGp, and EXp, are called the existence formulas since their semantics are all involved in "there exists at least one path." Each of these four existence formulas is related to one of the universal formulas, which is summarized in Table 3.

Comparing A$\neg p$ $\tilde{\mathrm{U}}$ $\neg q$ and $\varphi_1 = \neg p$ $\tilde{\mathrm{U}}$ $\neg q$, it can be observed that these two formulas have the same semantics. Thus, $\neg\varphi_1 =$ EpUq. Therefore, the algorithm TL-MC-DNA(DNACODE(A), DNACODE(A($f'' = \varphi_1$))) can be used to check the CTL formula EpUq. Similarly, the algorithm TL-MC-DNA(DNACODE(A), x) can be employed to check the CTL formulas EFp, EGp, and EXp. The detailed algorithm is formulated as shown in Algorithm 2. It should be noted that when a negative form of an atomic proposition occurs in the algorithm and is assigned as its argument, only one new atomic proposition is needed in the design of DNA encoding. No modification is needed on the algorithm, the FSA structure, or the encoding scheme of sticker automata.

3.1.3. The DNA Model Checking for the Basic CTL Formulas. The principle of this algorithm is as follows. (1) If a basic CTL formula is a universal formula, Algorithm 1 will be called. (2) And if a basic CTL formula is an existence formula, Algorithm 2 will be called. In this way, model checking of the basic CTL formulas can be conducted. The algorithm is formulated as shown in Algorithm 3.

3.1.4. Complexity Analysis. The time complexity of the algorithm TL-MC-DNA is $O(m + n)$ [55], where m means the

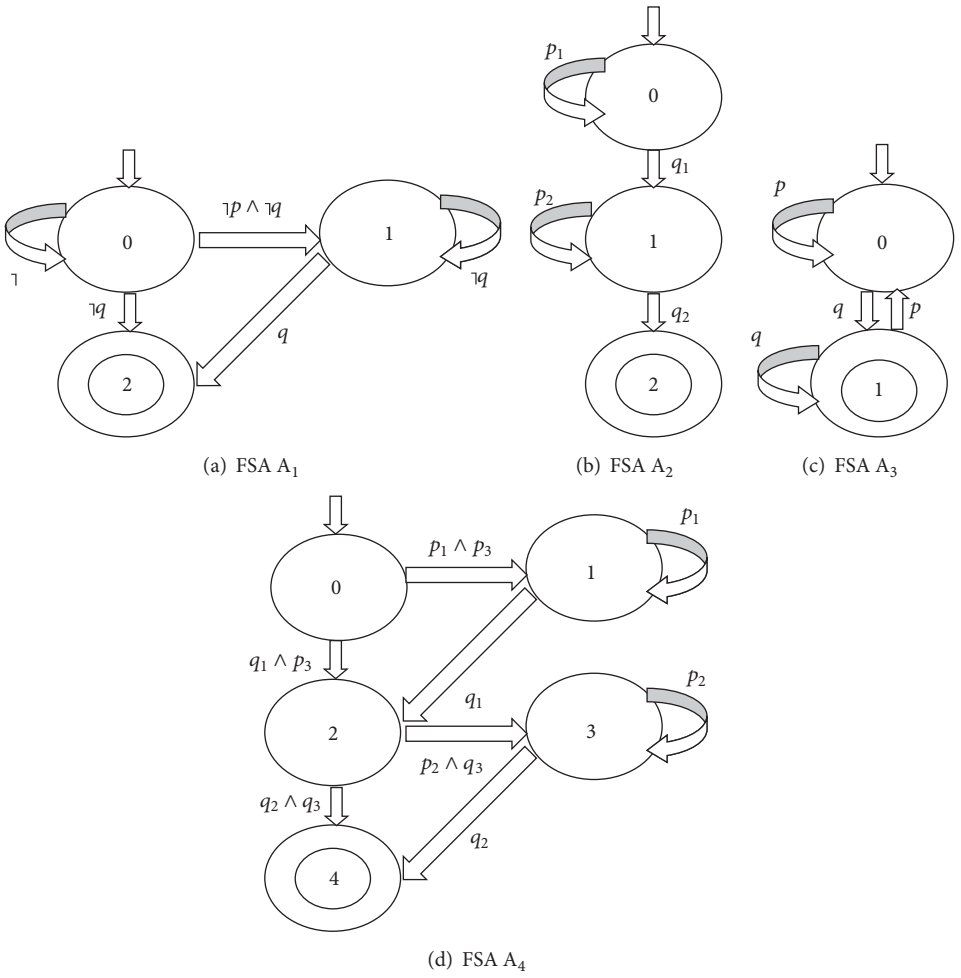


Figure 9: The four FSA models of the four formulas.

INPUT: The encoding of one sticker automata for an FSA of a system **A** and the encoding of the other sticker automaton for an FSA of a universal CTL formula $\mathbf{f_q}$, 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
$g \coloneqq p$Uq // where g is a f formula
CASE AFp
$g \coloneqq$ Fp // where g is a f formula
CASE AGp
$g \coloneqq$ Gp // where g is a f formula
CASE AXp
$g \coloneqq$ Xp // where g is a f formula
ENDSELECT
Step 2: $y \coloneqq$ TL-MC-DNA(DNACODE(A), DNACODE(A(g)))
Step 3: IF y = "yes", THEN return "yes", ELSE return "no"
END

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

INPUT: The encoding of one sticker automata for an FSA **A** of a system and the encoding of the other sticker automaton for an FSA of an existence CTL formula $\mathbf{f_c}$, where $f_c = \mathrm{E}p\mathrm{U}q$, $\mathrm{EF}p$, $\mathrm{EG}p$ or $\mathrm{EX}p$
OUTPUT: whether A satisfies f_c, or not
BEGIN
SELECT CASE f_c
  CASE $\mathrm{E}p\mathrm{U}q$
    Step 1: $g \coloneqq \varphi_1$
    Step 2: $y \coloneqq$ TL-MC-DNA(DNACODE(A), DNACODE(A(g))) // where g is a f'' formula
    Step 3: IF y = "yes", THEN return "no", ELSE return "yes" //$\varphi_1 = \neg(\mathrm{E}p\mathrm{U}q)$
  CASE $\mathrm{EF}p$
    Step 1: $g \coloneqq \mathrm{G}\neg p$
    Step 2: $y \coloneqq$ TL-MC-DNA(DNACODE(A), DNACODE(A(g))) // where g is a f formula
    Step 3: IF y = "yes", THEN return "no", ELSE return "yes" // $\mathrm{G}\neg p = \neg(\mathrm{EF}p)$
  CASE $\mathrm{EG}p$
    Step 1: $g \coloneqq \mathrm{F}\neg p$
    Step 2: $y \coloneqq$ TL-MC-DNA(DNACODE(A), DNACODE(A(g))) // where g is a f formula
    Step 3: IF y = "yes", THEN return "no", ELSE return "yes" // $\mathrm{F}\neg p = \neg(\mathrm{EG}p)$
  CASE $\mathrm{EX}p$
    Step 1: $g \coloneqq \mathrm{X}\neg p$
    Step 2: $y \coloneqq$ TL-MC-DNA(DNACODE(A), DNACODE(A(g))) // where g is a f formula
    Step 3: IF y = "yes", THEN return "no", ELSE return "yes" // $\mathrm{X}\neg p = \neg(\mathrm{EX}p)$
ENDSELECT
END

Algorithm 2: CTLC-MC-DNA(DNACODE(A), DNACODE(A(f_c))), the DNA model checking algorithm for the existence CTL formulas.

INPUT: The encoding of one sticker automata for an FSA **A** of a system and the encoding of the other sticker automaton for an FSA of a basic CTL formula $\mathbf{f_{CTL}}$
OUTPUT: whether A satisfies f_{CTL}, or not
BEGIN
  Step 1: IF there exists f_c, such that $f_{\mathrm{CTL}} = f_c$, THEN call CTLC-MC-DNA(DNACODE(A), DNACODE(A(f_c)))
   ELSEIF there exists f_q, such that $f_{\mathrm{CTL}} = f_q$, THEN call CTLQ-MC-DNA(DNACODE(A), DNACODE(A(f_q)))
END

Algorithm 3: CTL-MC-DNA(DNACODE(A), DNACODE(A(f_{CTL}))), the DNA model checking algorithm for the basic CTL formulas.

Table 3: The relationships between the existence and the universal CTL formulas.

Existence formulas	Universal formulas	Relationships
$\mathrm{E}p\mathrm{U}q$	$\mathrm{A}p\mathrm{U}q$	$\neg\mathrm{E}p\mathrm{U}q = \mathrm{A}\neg p \,\underline{\underline{\mathrm{U}}}\, \neg q$
		$\mathrm{E}p\mathrm{U}q = \neg\mathrm{A}\neg p \,\underline{\underline{\mathrm{U}}}\, \neg q$
$\mathrm{EG}p$	$\mathrm{AF}p$	$\neg\mathrm{EG}p = \mathrm{AF}\neg p$
		$\mathrm{EG}p = \neg\mathrm{AF}\neg p$
$\mathrm{EF}p$	$\mathrm{AG}p$	$\neg\mathrm{EF}p = \mathrm{AG}\neg p$
		$\mathrm{EF}p = \neg\mathrm{AG}\neg p$
$\mathrm{EX}p$	$\mathrm{AX}p$	$\neg\mathrm{EX}p = \mathrm{AX}\neg p$
		$\mathrm{EX}p = \neg\mathrm{AX}\neg p$

number of nodes in an automaton and n means the number of edges in this automaton. Therefore, Algorithm 1 needs to execute $O(m + n) + O(3) = O(m + n)$ times operations. Similarly, Algorithm 2 needs to execute $O(m + n) + O(3) = O(m + n)$ times operations. Algorithm 3 calls Algorithm 1 or Algorithm 2, so that the complexity of Algorithm 3 is $O(m + n)$. In comparison, the model checking of the basic CTL formulas based on classical computing has a square complexity.

Regarding the efficiency of the algorithm in the classical model checking based on electronic computing, a computational process will advance sequentially. In the DNA model checking, the process is different. A large number of molecules execute computations at the same time, in a parallel manner. Although/since the massive computational units (i.e., molecules) are involved in computation, the efficiency of the algorithm is improved. In contrast, the classical model checking requires fewer computational units but more computational steps. In short, the DNA computing is better in the time at the cost of space, compared with the classical computing. Thus, the two kinds of computing approaches are complementary.

3.2. The DNA Model Checking for the Basic ITL Formulas

3.2.1. The DNA Model Checking for the Basic ITL Formulas. There are two basic ITL formulas. The basic ITL formula $(p_1\mathrm{U}q_1);(p_2\mathrm{U}q_2)$ can perform DNA model checking by calling the algorithm TL-MC-DNA(DNACODE(A),

INPUT: The encoding of one sticker automata for an FSA **A** of a system and the encoding of the other sticker automaton for an FSA of a basic ITL formula $\mathbf{f_{ITL}}$
OUTPUT: whether A satisfies f_{ITL}, or not
BEGIN
Step 1: IF $f_{\mathrm{ITL}} = \varphi_2$, THEN call TL-MC-DNA(DNACODE(A), DNACODE(A($f'' = \varphi_2$)))
ELSEIF $f_{\mathrm{ITL}} = \varphi_3$, THEN call TL-MC-DNA(DNACODE(A), DNACODE(A($f'' = \varphi_3$)))
END

Algorithm 4: ITL-MC-DNA(DNACODE(A), DNACODE(A(f_{ITL}))), the DNA model checking algorithm for the basic ITL formulas.

INPUT: The encoding of one sticker automata for an FSA **A** of a system and the encoding of the other sticker automaton for an FSA of a basic PTL formula $\mathbf{f_{PTL}}$
OUTPUT: whether A satisfies f_{PTL}, or not
BEGIN
Step 1: IF $f_{\mathrm{PTL}} = \varphi_4$, THEN call TL-MC-DNA(DNACODE(A), DNACODE(A($f'' = \varphi_4$)))
END

Algorithm 5: PTL-MC-DNA(DNACODE(A), DNACODE(A(f_{PTL}))), the DNA model checking algorithm for the basic PTL formulas.

DNACODE(A($f'' = \varphi_2$))). The basic ITL formula $(p\mathrm{U}q)^*$ can perform DNA model checking by calling the algorithm TL-MC-DNA(DNACODE(A), DNACODE(A($f'' = \varphi_3$))). The algorithm is formulated as shown in Algorithm 4.

3.2.2. Complexity Analysis. Algorithm 4 calls the algorithm TL-MC-DNA, which has a complexity of $O(m + n)$ [55]. Therefore, the complexity of Algorithm 4 is $O(m + n)$. In comparison, the model checking of the basic ITL formulas based on classical computing has an exponential complexity.

3.3. The DNA Model Checking for the Basic PTL Formula

3.3.1. The DNA Model Checking for the Basic PTL Formula. The DNA model checking for the basic PTL formula $((p_1\mathrm{U}q_1), (p_2\mathrm{U}q_2))$ prj $(p_3 \wedge \mathrm{X}q_3)$ can be performed by calling the algorithm TL-MC-DNA(DNACODE(A), DNACODE(A($f'' = \varphi_4$))). The algorithm is formulated as shown in Algorithm 5.

3.3.2. Complexity Analysis. Algorithm 5 calls the algorithm TL-MC-DNA which has a complexity of $O(m+n)$ [55]. Thus, the complexity of Algorithm 5 is $O(m+n)$. In comparison, the model checking of the basic PTL formula based on classical computing has an exponential complexity.

4. Simulated Experiments

The core implement component of our new approaches is TL-MC-DNA algorithm which is called by all the new methods. We have implemented this algorithm on the general model of sticker automata, with a simulation platform called NUPACK [57]. It has been confirmed that, (1) for the nine FSAs of the nine specific temporal logic formulas, the algorithm TL-MC-DNA can be realized effectively in molecular biology; (2) for the above FSAs, one can design their appropriate encoding of sticker automata, so that the accuracy rate of base pairing reaches more than 99% [55]. For the four FSAs of the formulas presented in Section 2.7, it is important to implement the TL-MC-DNA algorithm effectively in molecular biology. In particular, the biological effectiveness of the algorithms from 1 to 5 is dependent on this. Therefore, the same experimental platform and experimental means with the ones in [55] are employed to carry out the molecular biological simulated experiments.

The design of the DNA encoding is in relation to the success of the experiment. In order to ensure the specificity of hybridization, an encoding sequence must satisfy some physical constraints and thermodynamic constraints [58]. In this paper, the thermodynamic constraints, including the thermal denaturizing temperature, and the free energy are studied only because the problem is limited by the physical constraints [55]. NUPACK can be employed to design the DNA encodings for sticker automata, and this tool can simulate the hybridization phenomena which originate from the running of the TL-MC-DNA algorithm. This experimental way has been proved to be scientific in [55].

Experimental Procedure. (1) According to Figures 7 and 9, one can design the encoding of the sticker automata for systematic FSAs shown in each subgraph, as well as the encoding of the sticker automata for FSAs of formulas shown in each subgraph, respectively; (2) for these FSAs mentioned above, one can simulate the process of hybridization between some single-stranded DNA molecules; (3) according to the five algorithms proposed in this paper, one can get the results of model checking of various temporal logic formulas, by reading the results of hybridization.

Experimental Objective. The objective is to test the correctness, effectiveness, and biological reliability of the new algorithms.

Table 4: Checking for φ_1: the designed encoding sequence, where WC means Watson-Crick complementary strand of code.

Code	5′ GCCAGAATTGCAAGGCAGCGAATTGCAAGGCGCGGAATTGCAAGGCCCCGAATTGCAAGGCCGTCCGACGC 3′
WC	3′ CGGTCTTAACGTTCCGTCGCTTAACGTTCCGCGCCTTAACGTTCCGGGGCTTAACGTTCCGGCAGGCTGCG 5′

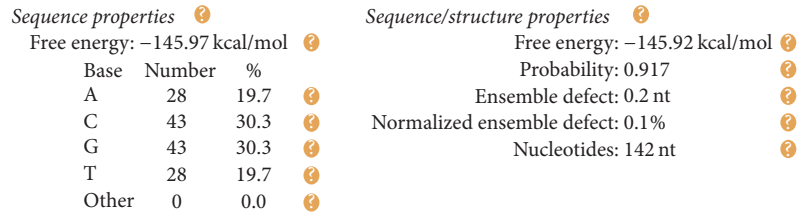
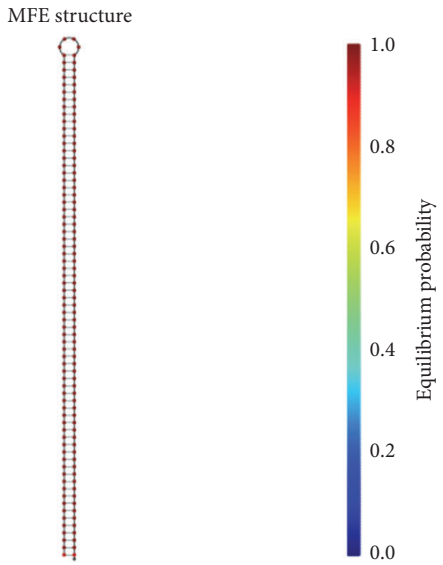
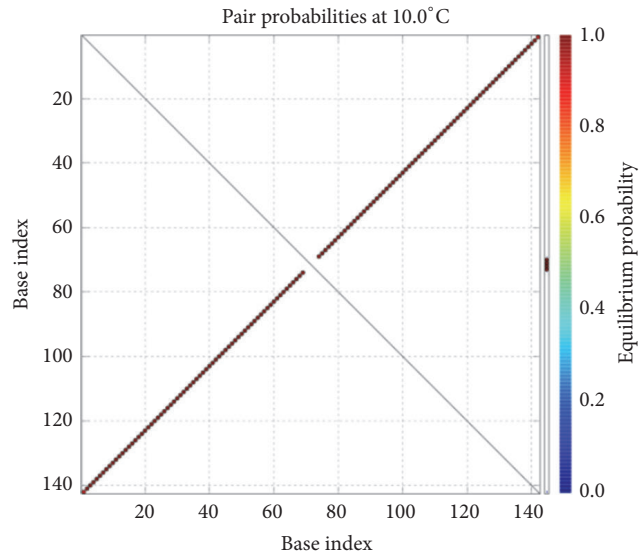


Figure 10: Checking the formula φ_1: the structural properties of encoding sequence.



Free energy of secondary structure: −145.92 kcal/mol

Figure 11: Thermodynamic analysis for φ_1: minimum free energy structure.



Free energy of strand ($-kT$ log Q): −145.97 kcal/mol

Figure 12: Checking for φ_1: pairing probability in equilibrium.

4.1. Simulated Experiments for φ_1

4.1.1. Encoding Designs. The DNA encoding via NUPACK is designed, as illustrated in Table 4. Figures 10, 11, and 12 show the thermodynamic analysis of the encoding sequence at 10°C. As shown in Figure 10, the Normalized Ensemble Defect (NED) means the incorrect matching ratio of the nucleotides when a biochemical reaction is in equilibrium. 0% implies an optimal design, whereas 100% implies the worst design. The NED of our coding sequence is 0.1%.

The principle of the minimum free energy points out that the free energy is minimized when a biochemical reaction is in equilibrium. As shown in Figure 11, the color of the match between two kinds of molecules is dark red. The probability of the following event almost reaches 100%: the double-stranded molecule is completely matched. We find this fact by comparing color changes of the vertical bar that indicate the balance probability. Thus, its free energy is approximately equal to the minimum free energy.

As shown in Figure 12, the position of the red line indicates that all bases in the two single strands are completely complementary to each other, and the color of the red line indicates that the probability of all the pairs is approximately equal to 1. As analyzed above, our DNA sequence satisfies the minimum free energy constraint and the DNA molecules that participate in the reaction have a basically consistent temperature of solution chain. Therefore, the experimental results obtained from this encoding are reliable and effective in biology.

In fact, Table 4 indicates the encoding rules for the input strings, as shown in Table 5. According to Table 5 and the principle of encoding of sticker automata, we can deduce the encoding of the sticker automaton characterizing φ_1, as shown in Table 6.

4.1.2. Simulated Experiments. With the DNA code given in Section 4.1.1 at hand, we can conduct our simulated experiments. It should be noted that, in Section 4.1.2, all the

Table 5: Checking for φ_1: the encoding rules of input strings characterizing runs, encoding by the way of sticker automata.

Object of code	DNA code
Initiator sequence	$I_1 = 5'$ GCCA $3'$
Spacer sequence	$X_0 = 5'$ GAA $3'$, $X_1 = 5'$ TTG $3'$, $X_2 = 5'$ CAA $3'$, $X_3 = 5'$ GGC $3'$
Terminator sequence	$I_2 = 5'$ CGTC $3'$
Atomic proposition	$p = 5'$ CGA $3'$, $q = 5'$ CCC $3'$, $r = \neg p = 5'$ CGC $3'$, $s = \neg q = 5'$ AGC $3'$, $u = r \wedge s = 5'$ GCG $3'$

Table 6: Checking for φ_1: the encoding of FSA A_1 of formula, encoding by the way of sticker automata, where sto() means WC.

Object of code	Abbreviated transition rule	DNA code
Initial state s_0	None	$3'$ sto($I_1\ X_0$) $5' = 3'$ CGGTCTT $5'$
Acceptance state s_2	None	$3'$ sto($X_3\ I_2$) $5' = 3'$ CCGGCAG $5'$
Transition rule $t(s_0, s) = s_0$	$t0s0$	$3'$ sto($X_1X_2X_3\ s\ X_0$) $5' = 3'$ AACGTTCCGTCGCTT $5'$
Transition rule $t(s_0, u) = s_1$	$t0u1$	$3'$ sto($X_1X_2X_3\ u\ X_0X_1$) $5' = 3'$ AACGTTCCGCGCCTTAAC $5'$
Transition rule $t(s_0, s) = s_2$	$t0s2$	$3'$ sto($X_1X_2X_3\ s\ X_0X_1X_2$) $5' = 3'$ AACGTTCCGTCGCTTAACGTT $5'$
Transition rule $t(s_1, s) = s_1$	$t1s1$	$3'$ sto($X_2X_3\ s\ X_0X_1$) $5' = 3'$ GTTCCGTCGCTTAAC $5'$
Transition rule $t(s_1, q) = s_2$	$t1q2$	$3'$ sto($X_2X_3\ q\ X_0X_1X_2$) $5' = 3'$ GTTCCGGGGCTTAACGTT $5'$

Table 7: The runs of the system M_1.

Path	DNA code of the path or sequence of nodes (atomic propositions) crossed by the path
Code of path 1	GCCA GAATTGCAAGGC AGC GAATTGCAAGGC AGC | GCG GAATTGCAAGGC CCC GAATTGCAAGGC CGTC
Sequence of nodes crossed by path 1	$0, 1, 2\ (s, s \mid u, q)$
Code of path k	GCCA GAATTGCAAGGC (AGC GAATTGCAAGGC AGC | GCG GAATTGCAAGGC)^k CCC GAATTGCAAGGC CGTC
Sequence of nodes crossed by path k	$(0, 1)^k, 2$

encoding of the DNA molecules is written from left to right with a $5'$-$3'$ direction, which is consistent with the way of writing in NUPACK.

We will check whether or not the systematic FSA M_1 satisfies the formula φ_1. According to the DNA codes given by Section 4.1.1, we can get all the paths which come from the systematic runs, as shown in Table 7, where k is a natural number. The transition rules shown in Table 6 clearly indicate that none of the atomic proposition excerpts for s, u, and q takes part in the transitions of states. Therefore, we do no need to consider whether or not the states satisfy the atomic propositions p and r.

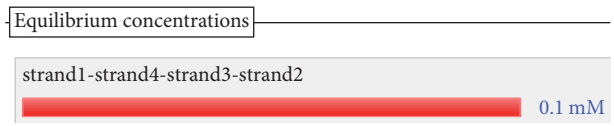
First, we will check path 1. There are two possible runs in this path. Without loss of generality, we support that the atomic proposition sequence which is crossed by the run is suq.

All the molecules expressing the runs begin with GCCA-GAA and end with GGCCGTC. Thus, we only need to consider d = TTGCAAGGCAGCGAATTGCAAGGCGCG GAATTGCAAGGCCCCGAATTGCAA. In short, we will observe whether or not hybridization occurs between the DNA molecules expressing transitions and the molecule d. For this experiment, the following six kinds of molecules are poured into a container with a volume of 10^{-15} L: d, $t0s0$, $t0u1$, $t0s2$, $t1s1$, and $t1q2$, for observing the hybridization.

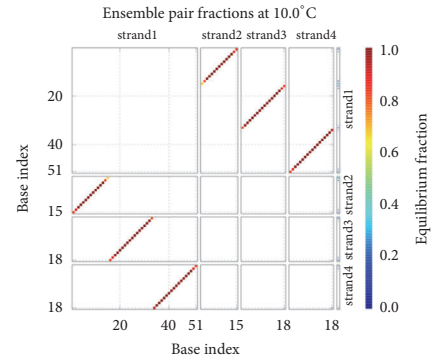
The systematic run, which is expressed by the molecule d, crosses the three states. If hybridization occurs between the DNA molecules expressing transitions and the molecule d, there are not more than three kinds of molecules which are the WC of some segment of d, involved in the specific hybridization. For selecting three kinds of molecules from all the five kinds of WC molecules, one has ten choices. Thus, the following ten groups of subexperiments are performed, accordingly.

(1) Group 1: t0s0, t0u1, t1q2, and d. The concentrations of the four kinds of molecules are all 100 uM, and their molecular numbers are all 60000. With the temperature naturally dropped to 10°C, the hybridization reaction is observed. Figures 13(a) and 13(b) show the result of the hybridization, where strand1, strand2, strand3, and strand4 mean d, $t0s0$, $t0u1$, and $t1q2$, respectively.

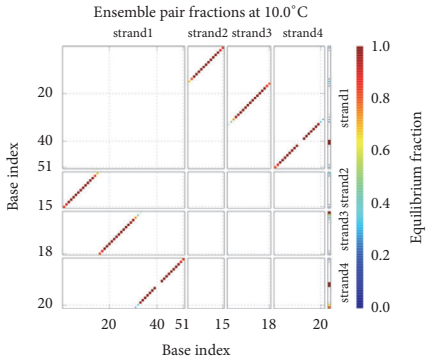
In Figure 13(b), the coordinates of the location of the first red line from top to bottom indicate that the base sequence of the molecule d from the 1st to the 15th sites at $5'$-$3'$ direction is paired with all of the fifteen bases of the molecule $t0s0$ at $3'$-$5'$ direction. The coordinates of the location of the second red line from top to bottom indicate that the base sequence of the molecule d from the 16th to the 33rd sites at $5'$-$3'$ direction is paired with all of the eighteen bases of the molecule $t0u1$ at $3'$-$5'$ direction. The coordinates of the location of the third



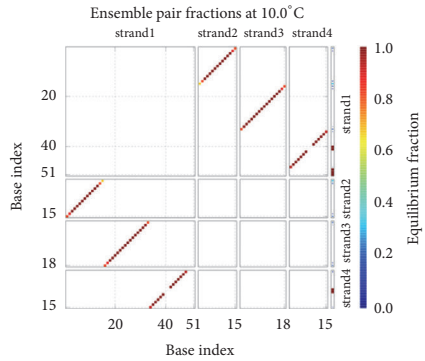
(a) Group 1: molecular concentrations



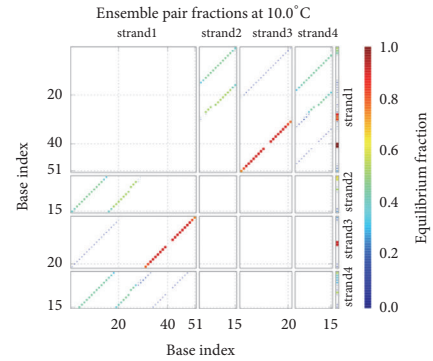
(b) Group 1: location and rate of pairing



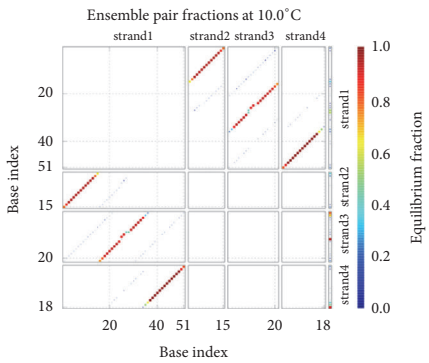
(c) Group 2: location and rate of pairing



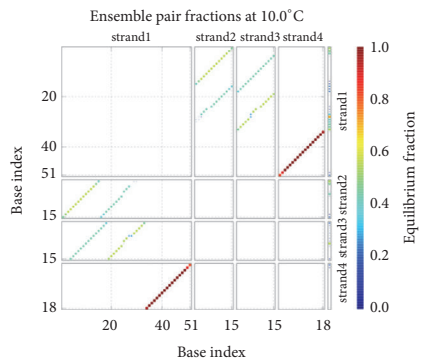
(d) Group 3: location and rate of pairing



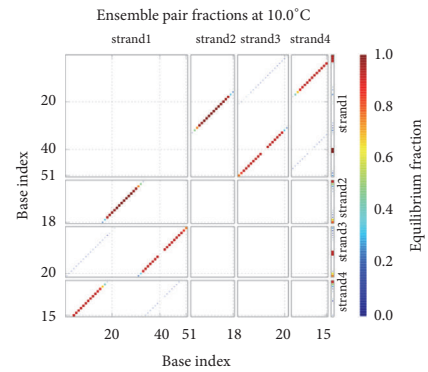
(e) Group 4: location and rate of pairing



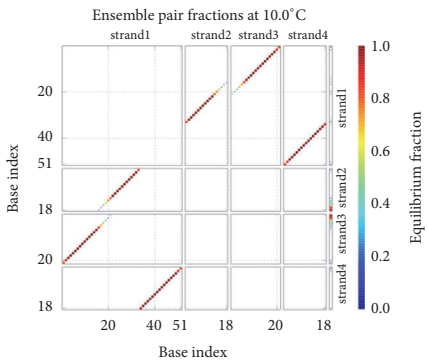
(f) Group 5: location and rate of pairing



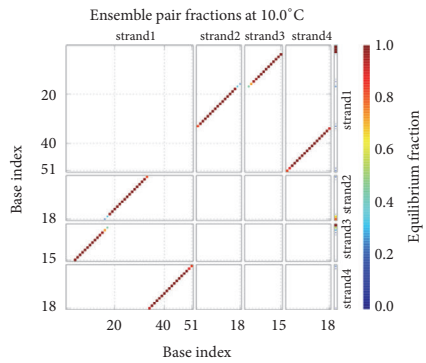
(g) Group 6: location and rate of pairing



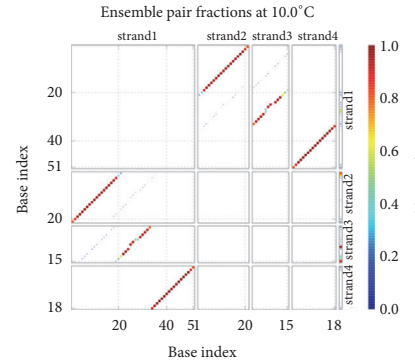
(h) Group 7: location and rate of pairing



(i) Group 8: location and rate of pairing



(j) Group 9: location and rate of pairing



(k) Group 10: location and rate of pairing

Figure 13: Checking for φ_1: the groups of subexperimental results on base pairing and hybridization.

red line from top to bottom indicate that the base sequence of the molecule d from the 34th to the 51st sites at $5'$-$3'$ direction is paired with all of the eighteen bases of the molecule $t1q2$ at $3'$-$5'$ direction. The results suggest that the complete double-stranded DNA molecules are formed, and the hybridization among the four kinds of single-stranded DNA molecules is specific.

Comparing the color of the three red lines with the color change of the vertical bar on the right side of Figure 13(b), it can be clearly seen that the former colors are very close to the color at the top of the vertical bar. This phenomenon suggests that the probabilities of these base pairs are close to 100%. This is a higher degree of specificity.

As shown in Figure 13(a), the concentration of the molecule strand1-strand2-strand3-strand4 is 100 uM, and the concentrations of the molecules $t0s0$, $t0u1$, $t1q2$, and d are approximately equal to 0 after their hybridization. This indicates that all of the molecular reactants are involved in the specific hybridization, due to 100 uM/100 uM = 100%. Therefore, both the false negative rate and the false positive rate are approximate to 0. The true positive rate is approximately equal to 100%. In short, the results show that the four kinds of molecules are hybridized with strong specificity.

(2) Group 2: t0s0, t0u1, t0s2, and d. The concentrations of the four kinds of molecules are all 100 uM, and their molecular numbers are all 60000. As the temperature naturally drops to 10°C, the hybridization reaction is observed. Figure 13(c) shows the result of the hybridization, where strand1, strand2, strand3, and strand4 mean d, $t0s0$, $t0u1$, and $t0s2$, respectively.

See Figure 13(c). There exists a red dot in the segment of strand1 of the vertical thin bar on the right side of strand4, indicating that some bases of strand1 are not paired with others. The results suggest that the four kinds of molecules in group 2 do not form complete double strands.

For all the other groups, all of the biochemical conditions and processes are similar to the groups above.

(3) Group 3: t0s0, t0u1, t1s1, and d. Figure 13(d) shows the result. There exist some red dots in the segment of strand1 of the vertical thin bar on the right side of strand4, suggesting that the four kinds of molecules do not form complete double strands.

(4) Group 4: t0s0, t0s2, t1s1, and d. Figure 13(e) shows the results. No red line is found at the $5'$ end of strand1, indicating that the $5'$ end of strand1 is not paired with any molecule. This suggests that the four kinds of molecules do not form complete double strands.

(5) Group 5: t0s0, t0s2, t1q2, and d. The results are shown in Figure 13(f). There exist some red dots in the segments of strand3 and strand4 of the vertical thin bar on the right side of strand4, suggesting that the four kinds of molecules do not form complete double strands.

(6) Group 6: t0s0, t1s1, t1q2, and d. As shown in Figure 13(g), no red line is found at the $5'$ end of strand1, suggesting that the four kinds of molecules do not form complete double strands.

Table 8: The results: checking for φ_1 in the different paths of M_1 (whether or not the path satisfies φ_1).

Formula	Path 1	Path k, where $15 > k > 1$	Path 15	Does M_1 satisfy φ_1?
φ_1	Yes	Yes	Yes	Yes

(7) Group 7: t0u1, t0s2, t1s1, and d. As shown in Figure 13(h), there are some red dots in the segments of strand1 of the vertical thin bar on the right side of strand4, suggesting that the four kinds of molecules do not form complete double strands.

(8) Group 8: t0u1, t0s2, t1q2, and d. As shown in Figure 13(i), there are some red dots in the segments of strand2 and strand3 of the vertical thin bar on the right side of strand4, suggesting that the four kinds of molecules do not form complete double strands.

(9) Group 9: t0u1, t1s1, t1q2, and d. As shown in Figure 13(j), there exist a red dot in the segments of strand1 of the vertical thin bar on the right side of the strand4, suggesting that the four kinds of molecules do not form the complete double strands.

(10) Group 10: t0s2, t1s1, t1q2 and d. As shown in Figure 13(k), there exist some red dots in the segments of strand2 and strand3 of the vertical thin bar on the right side of strand4, suggesting that the four kinds of molecules do not form complete double strands.

According to the ten groups of subexperiments mentioned above, we find that only group 1 (i.e., $t0s0$, $t0u1$, $t1q2$, and d) can form complete double strands by the hybridization reaction. That is to say, the systematic run suq satisfies the formula φ_1, since the first state does not satisfy q, the second state satisfies none of p and q, and the third state satisfies q.

The above results are obtained when $k = 1$. It has been proved that a system satisfies the formula $p\mathrm{U}q$, if and only if all the runs whose lengths are less than $|V| * 2^{|V|-1} + |E|$ satisfy $p\mathrm{U}q$, where $|V|$ and $|E|$ mean the number of nodes and the number of edges in the systematic FSA, respectively [55]. Similarly, we can prove that this conclusion holds for φ_1. M_1 has three nodes and three edges. Thus, we need to check fifteen paths due to $k = 3 * 2^{3-1} + 3 = 15$. With the same experimental way, we have checked the kth path, as shown in Table 8. M_1 satisfies the formula φ_1 since all paths (i.e., runs) satisfy this formula.

By calling the procedure for checking φ_1, Algorithm 2 can get the model checking results on the formula $\mathrm{E}p\mathrm{U}q$. The model checking results on the eight basic CTL formulas are shown in Table 9. According to the experimental processes and results in Section 4.1, we can safely say that Algorithm 3, which can be employed to check the basic CTL formulas, has been effectively implemented in molecular biology.

Table 9: The model checking results: M_1 and the basic CTL formulas (whether or not the system M_1 satisfies these formulas).

Formula	Result	The used algorithm and decision basis
ApUq	No	TL-MC-DNA determines that M_1 does not satisfy pUq, and thus Algorithm 1 determines that M_1 does not satisfy ApUq
AFp	Yes	TL-MC-DNA determines that M_1 satisfies Fp, and thus Algorithm 1 determines that M_1 satisfies AFp
AGp	No	TL-MC-DNA determines that M_1 does not satisfy Gp, and thus Algorithm 1 determines that M_1 does not satisfy AGp
AXp	No	TL-MC-DNA determines that M_1 does not satisfy Xp, and thus Algorithm 1 determines that M_1 does not satisfy AXp
EpUq	No	Extended TL-MC-DNA determines that M_1 satisfies φ_1, and thus Algorithm 2 determines that M_1 does not satisfy EpUq
EFp	Yes	TL-MC-DNA determines that M_1 does not satisfy G$\neg p$, and thus Algorithm 2 determines that M_1 satisfies EFp
EGp	No	TL-MC-DNA determines that M_1 satisfies F$\neg p$, and thus Algorithm 2 determines that M_1 does not satisfy EGp
EXp	No	TL-MC-DNA determines that M_1 satisfies X$\neg p$, and thus Algorithm 2 determines that M_1 does not satisfy EXp

Table 10: Checking for φ_2 and φ_3: the designed encoding sequence.

Code	5′ CGCTCGAATCGGAATGGATCGAATCGGAATGATACGAATCGGAATGGAACGAATCGGAATGTTCCGAATCGG AATGTATCGAATCGGAATGTGACGAATCGGAATGCGGC 3′
WC	3′ GCGAGCTTAGCCTTACCTAGCTTAGCCTTACTATGCTTAGCCTTACCTTGCTTAGCCTTACAAGGCTTAGCCTT ACATAGCTTAGCCTTACACTGCTTAGCCTTACGCCG 5′

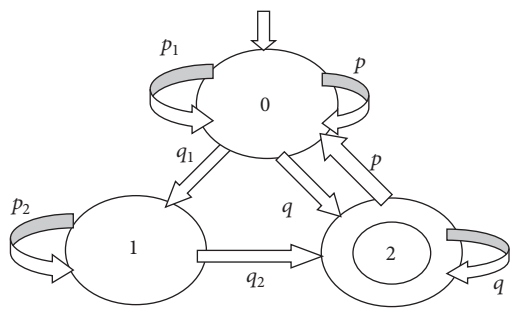


Figure 14: FSA of formula: merged graph A_5 of A_2 and A_3.

4.2. Simulated Experiments for φ_2 and φ_3

4.2.1. Encoding Designs. The formula φ_2 and the formula φ_3 need to be encoded with the same coding scheme since both formulas are ITL ones. Therefore, we combine the FSAs of these two formulas into one, as shown in Figure 14. Our design of a DNA encoding via NUPACK is shown in Table 10, while Figures 15, 16, and 17 show the thermodynamic analysis of the encoding sequence presented in Table 10 at 10°C. The NED of our coding sequence is 0.1%, which is illustrated in Figure 15. Its free energy is approximately equal to the minimum free energy, as shown in Figure 16. All bases in the two single strands are completely complementary to each other as shown in Figure 17, and the probabilities of all the pairs are approximately equal to 1. In Table 11, the encoding rules for the input strings are provided while Table 12 shows the encoding of the sticker automaton characterizing φ_2 and φ_3.

4.2.2. Simulated Experiments. With the DNA code given in Section 4.2.1 at hand, we can conduct our simulated experiments. It should be noted that, in Section 4.2.2, all the encoding of the DNA molecules is written from left to right with a 5′-3′ direction, which is consistent with the way of writing using NUPACK.

(1) Model Checking: Whether the Systematic FSA M_2 Satisfies φ_2 or Not. With our DNA codes, all the paths of M_2 are shown in Table 13, where k is a natural number. By observing the transition rules which are related to φ_2 and shown in Table 12, it can be seen that none of the atomic proposition excerpts for p_1, q_1, p_2, and q_2 takes part in the transitions of states. Therefore, we do no need to consider whether or not the states satisfy other atomic propositions.

First, we will check path 1. There are two possible runs in this path. Without loss of generality, we suppose that the atomic proposition sequence which is crossed by the run is $p_1q_1p_2q_2$. We only need to deal with d = ATCG GAATGGATCGAATCGGAATGATACGAATCGGAATGG AACGAATCGGAATGTTCCGAATCGGA. In short, we will observe whether or not hybridization occurs between the DNA molecules expressing transitions and the molecule d. To this end, we pour the following five kinds of molecules into a container with a volume of 10^{-15} L: d, $t0p_10$, $t0q_11$, $t1p_21$, and $t1q_22$, for observing the hybridization.

The concentrations of the five kinds of molecules reach 100 uM, and their molecular numbers are all 60000. With the temperature naturally dropped to 10°C, the hybridization reaction is observed. Figure 18 shows the result of the hybridization, where strand1, strand2, strand3, strand4, and strand5 mean d, t0$p_1$0, $t0q_11$, $t1p_21$, and $t1q_22$, respectively.

Table 11: Checking for φ_2 and φ_3: the encoding rules of input strings.

Object of code	DNA code
Initiator sequence	$I_1 = 5'$ CGCT $3'$
Spacer sequence	$X_0 = 5'$ CGA $3'$, $X_1 = 5'$ ATC $3'$, $X_2 = 5'$ GGA $3'$, $X_3 = 5'$ ATG $3'$
Terminator sequence	$I_2 = 5'$ CGGC $3'$
Atomic proposition	$p_1 = 5'$ GAT $3'$, $p_2 = 5'$ GAA $3'$, $q_1 = 5'$ ATA $3'$, $q_2 = 5'$ TTC $3'$, $p = 5'$ TAT $3'$, $q = 5'$ TGA $3'$

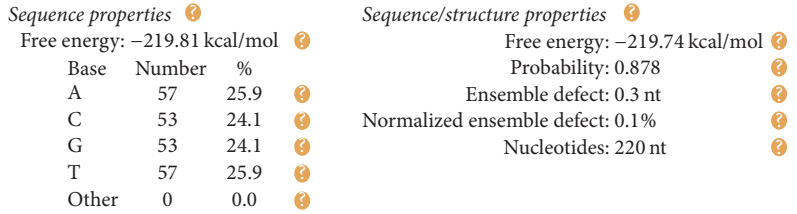


Figure 15: For φ_2 and φ_3: the structural properties of encoding sequence.

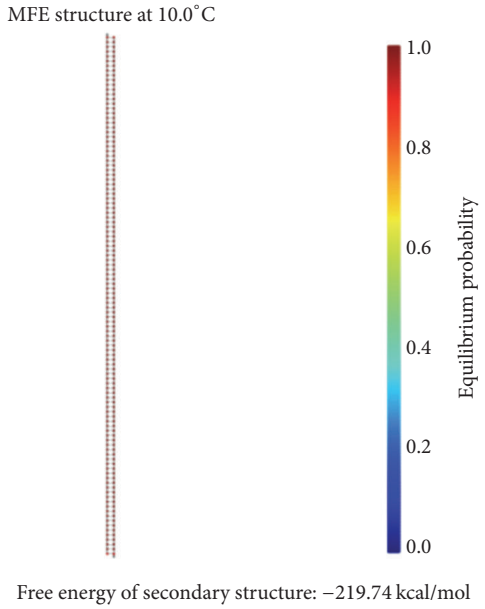


Figure 16: Thermodynamic analysis for φ_2 and φ_3: minimum free energy structure.

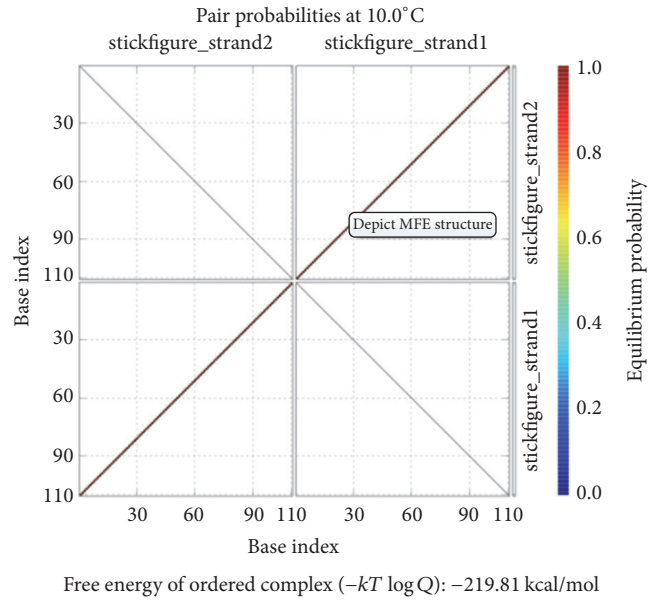
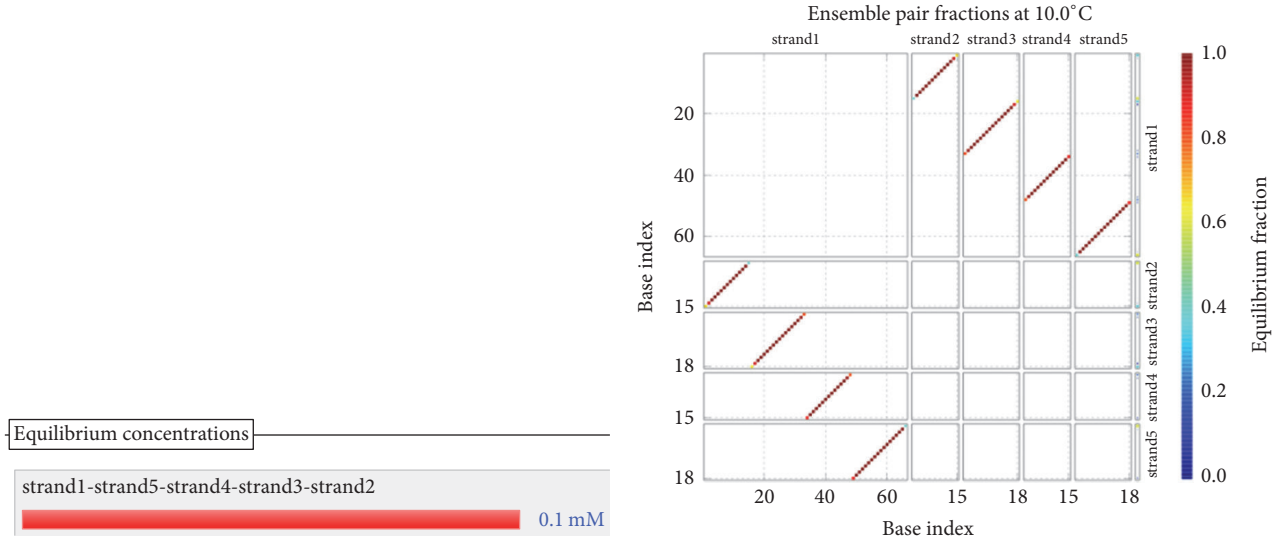


Figure 17: Checking for φ_2 and φ_3: pairing probability in equilibrium.

In Figure 18(b), the coordinates of the location of the four red lines from top to bottom indicate that the complete double-stranded DNA molecules are formed by the hybridization among the five kinds of single-stranded DNA molecules. Comparing the color of the four red lines with the color change of the vertical bar on the right side of Figure 18(b), we can see clearly that the probabilities of these base pairs are close to 100%. This is a higher degree of specificity. As shown in Figure 18(a), the concentration of the molecules indicates that the true positive rate is approximately equal to 100%. Once again, it suggests that the five kinds of molecules are hybridized with strong specificity. Thus, the systematic run $p_1q_1p_2q_2$ satisfies the formula φ_2.

The above results are gotten when $k = 1$. It has been proved that a system satisfies the formula pUq, if and only if all the runs whose lengths are less than $|V| * 2^{|V|-1} + |E|$ satisfy pUq, where $|V|$ and $|E|$ mean the number of nodes and the number of edges in the systematic FSA, respectively [55]. A system satisfies the formula φ_2, if and only if all the runs whose lengths are less than $(|V1| * 2^{|V1|-1} + |E1|) + (|V2| * 2^{|V2|-1} + |E2|)$ satisfy pUq since φ_2 is composed of the two pUq-like formulas sequentially, where $|V1|$ and $|E1|$ mean the number of nodes and the number of edges in the prefix interval of the systematic FSA, respectively, and $|V2|$ and $|E2|$ mean the number of nodes and the number of edges in the suffix interval of the systematic FSA, respectively. For M_2, $|V1| = 2$, $|E1| = 2$, $|V2| = 2$, and $|E2| = 1$. Thus, we need to check eleven paths due to $2 * 2^{2-1} + 2 + 2 * 2^{2-1} + 1 = 11$, as shown in Table 14. M_2 satisfies the formula φ_2 since all paths (i.e., runs) satisfy this formula. By calling the procedure for



(a) Molecular concentrations (b) Location and rate of pairing

FIGURE 18: Checking for φ_2: the experimental results on base pairing and hybridization.

TABLE 12: Checking for φ_2 and φ_3: the encoding of FSA A_5 of formula, where sto() means WC.

Object of code	Abbreviated transition rule	DNA code
Initial state s_0	None	$3'$ sto(I_1 X_0) $5' = 3'$ GCGA GCT $5'$
Acceptance state s_2	None	$3'$ sto(X_3 I_2) $5' = 3'$ TAC GCCG $5'$
Transition rule $t(s_0, p_1) = s_0$	$t0p_10$	$3'$ sto($X_1X_2X_3$ p_1 X_0) $5' = 3'$ TAG CCT TAC CTA GCT $5'$
Transition rule $t(s_0, p) = s_0$	$t0p0$	$3'$ sto($X_1X_2X_3$ p X_0) $5' = 3'$ TAG CCT TAC ATA GCT $5'$
Transition rule $t(s_0, q_1) = s_1$	$t0q_11$	$3'$ sto($X_1X_2X_3$ q_1 X_0X_1) $5' = 3'$ TAG CCT TAC TAT GCT TAG $5'$
Transition rule $t(s_0, q) = s_2$	$t0q2$	$3'$ sto($X_1X_2X_3$ q $X_0X_1X_2$) $5' = 3'$ TAG CCT TAC ACT GCT TAG CCT $5'$
Transition rule $t(s_1, p_2) = s_1$	$t1p_21$	$3'$ sto(X_2X_3 p_2 X_0X_1) $5' = 3'$ CCT TAC CTT GCT TAG $5'$
Transition rule $t(s_1, q_2) = s_2$	$t1q_22$	$3'$ sto(X_2X_3 q_2 $X_0X_1X_2$) $5' = 3'$ CCT TAC AAG GCT TAG CCT $5'$
Transition rule $t(s_2, p) = s_0$	$t2p0$	$3'$ sto(X_3 p X_0) $5' = 3'$ TAC ATA GCT $5'$
Transition rule $t(s_2, q) = s_2$	$t2q2$	$3'$ sto(X_3 q $X_0X_1X_2$) $5' = 3'$ TAC ACT GCT TAG CCT $5'$

checking φ_2, Algorithm 4 can get the model checking results on this basic ITL formula.

(2) Model Checking: Whether the Systematic FSA M_3 Satisfies φ_3 or Not. According to the DNA codes given by Section 4.2.1, we can get all the paths of M_3, as shown in Table 15, where k is a natural number. The transition rules, related to φ_3 and shown in Table 12, show that none of the atomic proposition excerpts for p and q takes part in the transitions of states. Therefore, we do no need to consider whether or not the states satisfy other atomic propositions.

First, we will check path 1. The atomic proposition sequence which is crossed by the run is pq. We only need to deal with d = ATCGGAATGTATCGAATCGGAATGT GACGAATCGGA. In short, we will observe whether or not hybridization occurs between the DNA molecules expressing transitions and the molecule d. To this end, we pour the following five kinds of molecules into a container with a volume of 10^{-15} L: d, $t0p0$, $t0q2$, $t2p0$, and $t2q2$, for observing the hybridization.

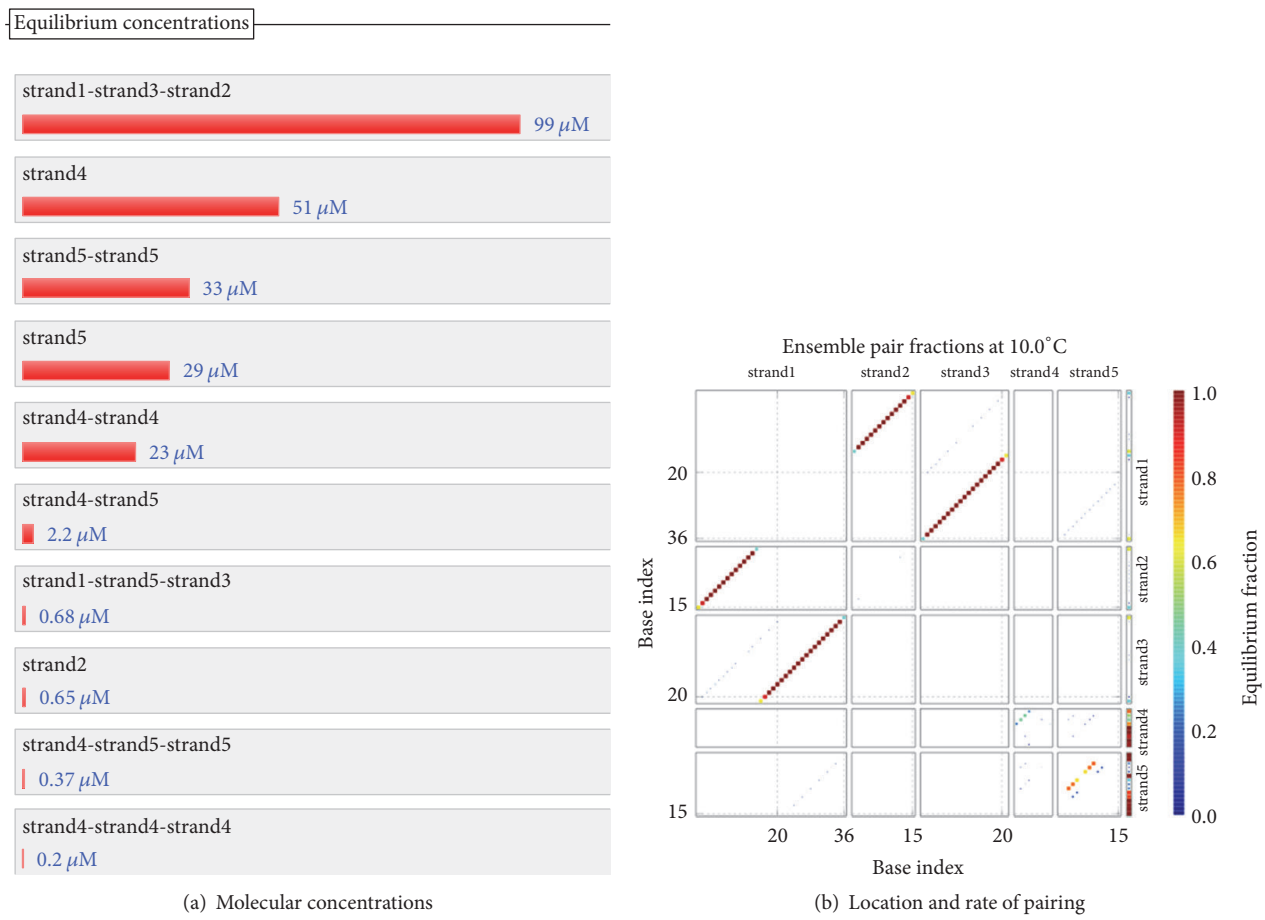
The concentrations of the five kinds of molecules reach 100 uM, and their molecular numbers are all 60000. The hybridization reaction is observed as the temperature naturally drops to 10°C. Figure 19 shows the result of the hybridization, where strand1, strand2, strand3, strand4, and strand5 mean d, $t0p0$, $t0q2$, $t2p0$, and $t2q2$, respectively.

As shown in Figure 19(b), the coordinates of the location of the two red lines from top to bottom indicate that the complete double-stranded DNA molecules are formed by the hybridization among $t0p0$, $t0q2$, and d. Comparing the color of the two red lines with the color change of the vertical bar on the right side of Figure 19(b), we can see clearly that the probabilities of these base pairs are close to 100%. This is a higher degree of specificity.

As shown in Figure 19(a), 99 uM/100 uM = 99% of the molecules d take part in the specific hybridization. Note that only the molecule strand1-strand3-strand2 is the product of the specific hybridization. Therefore, the true positive rate of the specific hybridization of d is approximately equal to 99%. Similarly, the false negative rate of d is equal to 0, and

Table 13: The runs of the system M_2.

Path	DNA code of the path or sequence of nodes (atomic propositions) crossed by the path
Code of path 1	CGCT CGAATCGGAATG GAT CGAATCGGAATG GAT | ATA CGAATCGGAATG GAA CGAATCGGAATG TTC CGAATCGGAATG CGGC
Sequence of nodes crossed by path 1	$0, 1, 2, 3\,(p_1, p_1 \mid q_1, p_2, q_2)$
Code of path k	CGCT CGAATCGGAATG (GAT CGAATCGGAATG GAT | ATA CGAATCGGAATG)^k GAA CGAATCGGAATG TTC CGAATCGGAATG CGGC
Sequence of nodes crossed by path k	$(0, 1)^k, 2, 3$



(a) Molecular concentrations

(b) Location and rate of pairing

Figure 19: Checking for φ_3: the experimental results on base pairing and hybridization.

Table 14: The results: checking for φ_2 in the different paths of M_2 (whether or not the path satisfies φ_2).

Formula	Path 1	Path k, where $11 > k > 1$	Path 11	Does M_2 satisfy φ_2?
φ_2	Yes	Yes	Yes	Yes

the false positive rate of d is 0.68 uM/100 uM = 0.68%. As for $t0p0$, its false negative rate is 0.65 uM/100 uM = 0.65%, its false positive rate is equal to 0, and its true positive rate is approximately equal to 99%. Similarly, the false negative rate of $t0q2$ is equal to 0, the false positive rate of $t0q2$ is 0.68 uM/100 uM = 0.68%, and the true positive rate of $t0q2$ is approximately equal to 99%. Once again, this suggests that the three kinds of molecules are hybridized with strong specificity. Thus, the systematic run pq satisfies the formula φ_3.

Similarly, the above results are gotten when $k = 1$. It also has been proved that a system satisfies the formula pUq, if and only if all the runs whose lengths are less than $|V| * 2^{|V|-1} + |E|$ satisfy pUq, where $|V|$ and $|E|$ mean the number of nodes and the number of edges in the systematic FSA, respectively [55]. As for φ_3, the same property holds, since φ_3 is composed of the one pUq-like formula recursively. For M_3, we need to check six paths due to $2 * 2^{2-1} + 2 = 6$, as shown in Table 16. M_3 satisfies the formula φ_3 since all paths (i.e., runs) satisfy this formula.

By calling the procedure for checking φ_3, Algorithm 4 can get the model checking results on this basic ITL formula.

Table 15: The runs of the system M_3.

Path	DNA code of the path or sequence of nodes (atomic propositions) crossed by the path
Code of path 1	CGCT CGAATCGGAATG TAT CGAATCGGAATG TGA CGAATCGGAATG CGGC
Sequence of nodes crossed by path 1	0, 1 (p, q)
Code of path k	CGCT CGAATCGGAATG (TAT CGAATCGGAATG TGA CGAATCGGAATG)^k CGGC
Sequence of nodes crossed by path k	$(0, 1)^k$

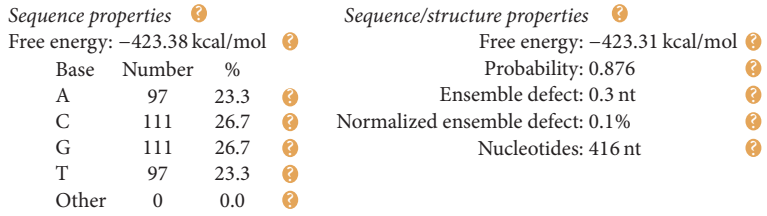


Figure 20: For φ_4: the structural properties of encoding sequence.

Table 16: The results: checking for φ_3 in the different paths of M_3 (whether or not the path satisfies φ_3).

Formula	Path 1	Path k, where $6 > k > 1$	Path 6	Does M_3 satisfy φ_3?
φ_3	Yes	Yes	Yes	Yes

According to the experimental processes and results in Section 4.2, we can safely say that Algorithm 4, which can be employed to check the basic ITL formulas, has been effectively implemented in molecular biology.

4.3. Simulated Experiments for φ_4

4.3.1. Encoding Designs. We have designed a DNA encoding via NUPACK, as shown in Table 17. Figures 20, 21, and 22 show the thermodynamic analysis of the encoding sequence presented in Table 17 at 10°C. As shown in Figure 20, the NED of our coding sequence is 0.1%. Figure 21 shows that its free energy is approximately equal to the minimum free energy and Figure 22 shows that all bases in the two single strands are completely complementary to each other, and the probabilities of all the pairs are approximately equal to 1. Table 18 gives the encoding rules for the input strings. And Table 19 shows the encoding of the sticker automaton characterizing φ_4.

4.3.2. Simulated Experiments. With the DNA code given in Section 4.3.1 at hand, we can conduct our simulated experiments. It should be noted that, in Section 4.3.2, all the encoding of the DNA molecules is written from left to right with a $5'$-$3'$ direction, which is consistent with the way of writing in NUPACK.

According to the DNA codes given by Section 4.3.1, we can get all the paths of M_2, as shown in Table 20. The transition rules related to φ_4 are shown in Table 19 and none of the atomic proposition excerpts for p_1, q_1, p_2, q_2, and m_1 takes part in the transitions of states. Therefore, we do no need to consider whether or not the states satisfy

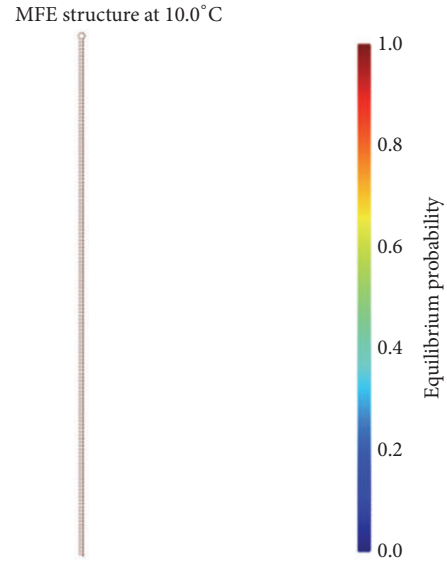


Figure 21: Thermodynamic analysis for φ_4: minimum free energy structure.

other atomic propositions. First, we will check path 1. There are two possible runs in this path. Without loss of generality, we support that the atomic proposition sequence which is crossed by the run is $m_1q_1p_2q_2$. We only need to deal with d = CGCATCATGTGGTCTTTGCATGGACG TAGTGATCGGCGCATCATGTGGTCTTTGCATGGACG TAATCCTCGGCGCATCATGTGGTCTTTGCATGGACG TACAAATCGGCGCATCATGTGGTCTTTGCATGGACG TAGGGATCGGCGCATCATGTGGTCTT.

In short, we will observe whether or not hybridization occurs between the DNA molecules expressing transitions and the molecule d. For selecting four kinds of molecules from all the eight kinds of WC molecules, one has seventy choices. Thus, we need to execute the seventy groups of subexperiments. For example, we pour the following five

Table 17: Checking for φ_4: the designed encoding sequence.

Code	5′ GCAGTCGGCGCATCATGTGGTCTTTGCATGGACGTAGTGATCGGCGCATCATGTGGTCTTTGCATGGACGTAAT CCTCGGCGCATCATGTGGTCTTTGCATGGACGTAAACGTCGGCGCATCATGTGGTCTTTGCATGGACGTAGGGAT CGGCGCATCATGTGGTCTTTGCATGGACGTAAACCCCGCCAAATTACATATGACCGACG 3′
WC	3′ CGTCAGCCGCGTAGTACACCAGAAACGTACCTGCATCACTAGCCGCGTAGTACACCAGAAACGTACCTGCATT AGGAGCCGCGTAGTACACCAGAAACGTACCTGCATTTGCAGCCGCGTAGTACACCAGAAACGTACCTGCATCCC TAGCCGCGTAGTACACCAGAAACGTACCTGCATTTGGGGCGGTTTAATGTATACTGGCTGC 5′

Table 18: Checking for φ_4: the encoding rules of input strings.

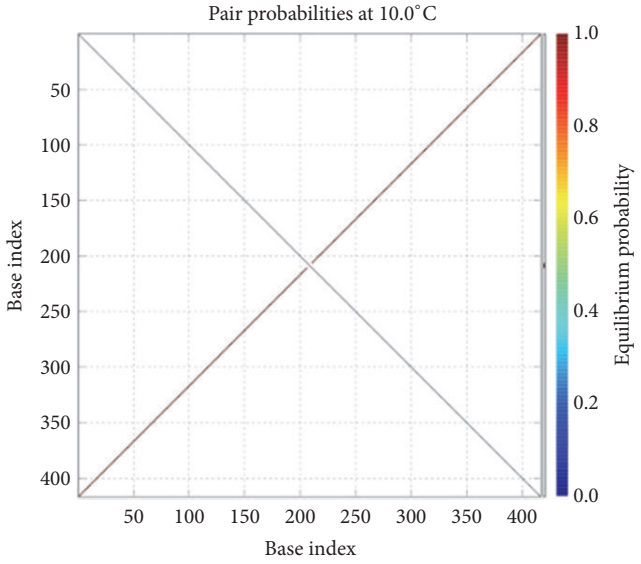
Object of code	DNA code (they are all in 5′-3′ direction from left to right)
Initiator sequence	I_1 = GCAG
Spacer sequence	X_0 = TCGG, X_1 = CGCA, X_2 = TCAT, X_3 = GTGG, X_4 = TCTT, X_5 = TGCA, X_6 = TGGA, X_7 = CGTA
Terminator sequence	I_2 = AACC
Atomic proposition	p_1 = CCGC, q_1 = ATCC, p_2 = CAAA, q_2 = GGGA, p_3 = TTAC, q_3 = ATAT $m_1 = p_1 \wedge p_3$ = GTGA, $m_2 = q_1 \wedge p_3$ = GACC, $m_3 = p_2 \wedge q_3$ = AACG, $m_4 = q_2 \wedge q_3$ = GACG

Table 19: Checking for φ_4: the encoding of FSA A_4 of formula, where sto() means WC.

Object of code	Abbreviated transition rule	DNA code (they are all in 5′-3′ direction from left to right)
Initial state s_0	None	sto(I_1 X_0) = CGTC AGCC
Acceptance state s_4	None	sto($X_5X_6X_7$ I_2) = ACGT ACCT GCAT TTGG
Transition rule $t(s_0, m_1) = s_1$	$t0m_11$	sto($X_1X_2X_3X_4X_5X_6X_7$ m_1 X_0X_1) = GCGT AGTA CACC AGAA ACGT ACCT GCAT CACT AGCC GCGT
Transition rule $t(s_0, m_2) = s_2$	$t0m_22$	sto($X_1X_2X_3X_4X_5X_6X_7$ m_2 $X_0X_1X_2$) = GCGT AGTA CACC AGAA ACGT ACCT GCAT CTGG AGCC GCGT AGTA
Transition rule $t(s_1, p_1) = s_1$	$t1p_11$	sto($X_2X_3X_4X_5X_6X_7$ p_1 X_0X_1) = AGTA CACC AGAA ACGT ACCT GCAT GGCG AGCC GCGT
Transition rule $t(s_1, q_1) = s_2$	$t1q_12$	sto($X_2X_3X_4X_5X_6X_7$ q_1 $X_0X_1X_2$) = AGTA CACC AGAA ACGT ACCT GCAT TAGG AGCC GCGT AGTA
Transition rule $t(s_2, m_3) = s_3$	$t2m_33$	sto($X_3X_4X_5X_6X_7$ m_3 $X_0X_1X_2X_3$) = CACC AGAA ACGT ACCT GCAT TTGC AGCC GCGT AGTA CACC
Transition rule $t(s_2, m_4) = s_4$	$t2m_44$	sto($X_3X_4X_5X_6X_7$ m_4 $X_0X_1X_2X_3X_4$) = CACC AGAA ACGT ACCT GCAT CTGC AGCC GCGT AGTA CACC AGAA
Transition rule $t(s_3, p_2) = s_3$	$t3p_23$	sto($X_4X_5X_6X_7$ p_2 $X_0X_1X_2X_3$) = AGAA ACGT ACCT GCAT GTTT AGCC GCGT AGTA CACC
Transition rule $t(s_3, q_2) = s_4$	$t3q_24$	sto($X_4X_5X_6X_7$ q_2 $X_0X_1X_2X_3X_4$) = AGAA ACGT ACCT GCAT CCCT AGCC GCGT AGTA CACC AGAA

Table 20: The runs of the system M_2.

Path	DNA code of the path or sequence of nodes (atomic propositions) crossed by the path
Code of path 1	GCAG TCGGCGCATCATGTGGTCTTTGCATGGACGTA CCGC | GTGA TCGGCGCATCATGTGGTCTTTGCATGGACGTA CCGC | ATCC TCGGCGCATCATGTGGTCTTTGCATGGACGTA CAAA TCGGCGCATCATGTGGTCTTTGCATGGACGTA GGGA TCGGCGCATCATGTGGTCTTTGCATGGACGTA AACC
Sequence of nodes crossed by path 1	0, 1, 2, 3 (only the atomic propositions that occur in the transition rules are retained: $p_1 \mid m_1, p_1 \mid q_1, p_2, q_2$)
Code of path k	GCAG TCGGCGCATCATGTGGTCTTTGCATGGACGTA (CCGC | GTGA TCGGCGCATCATGTGGTCTTTGCATGGACGTA CCGC | ATCC TCGGCGCATCATGTGGTCTTTGCATGGACGTA)^k CAAA TCGGCGCATCATGTGGTCTTTGCATGGACGTA GGGA TCGGCGCATCATGTGGTCTTTGCATGGACGTA AACC
Sequence of nodes crossed by path k	$(0, 1)^k$, 2, 3



Free energy of strand ($-kT$ log Q): −423.38 kcal/mol

FIGURE 22: Checking for φ_4: pairing probability in equilibrium.

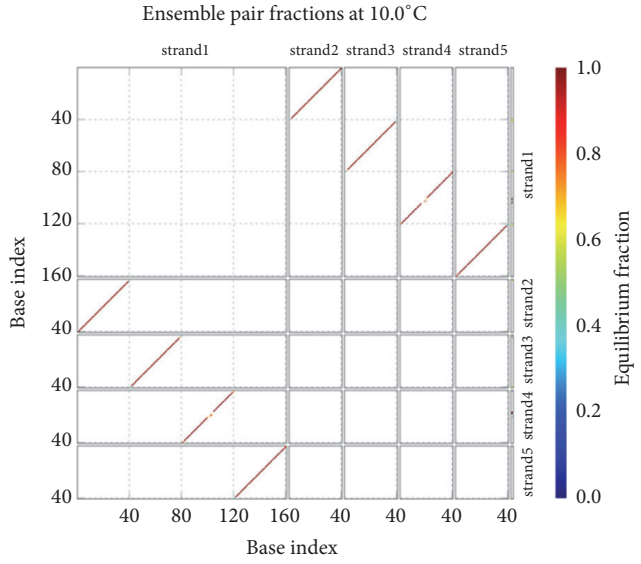


FIGURE 23: Checking for φ_4: a group of subexperimental results on hybridization: location and rate of base pairing.

kinds of molecules into a container with a volume of 10^{-15} L: d, $t0m_1 1$, $t1q_1 2$, $t2m_3 3$, and $t3q_2 4$, for observing the hybridization.

The concentrations of the five kinds of molecules reach 100 uM, and their molecular numbers are all 60000. With the temperature naturally dropped to 10°C, the hybridization reaction is observed. Figure 23 shows the result of the hybridization, where strand1, strand2, strand3, strand4, and strand5 mean d, $t0m_1 1$, $t1q_1 2$, $t2m_3 3$, and $t3q_2 4$, respectively.

As shown in the graph, the bases in the middle of strand4 are not paired with strand1, indicating that the complete double strands are not formed. As for any other group of subexperiments, the complete double strands are not formed. Thus, the systematic run $m_1 q_1 p_2 q_2$ does not satisfy the formula φ_4. Similarly, none of the runs in path 1 satisfies this formula. Therefore, there exists a path which does not satisfy φ_4. That is to say, M_2 does not satisfy the formula φ_4.

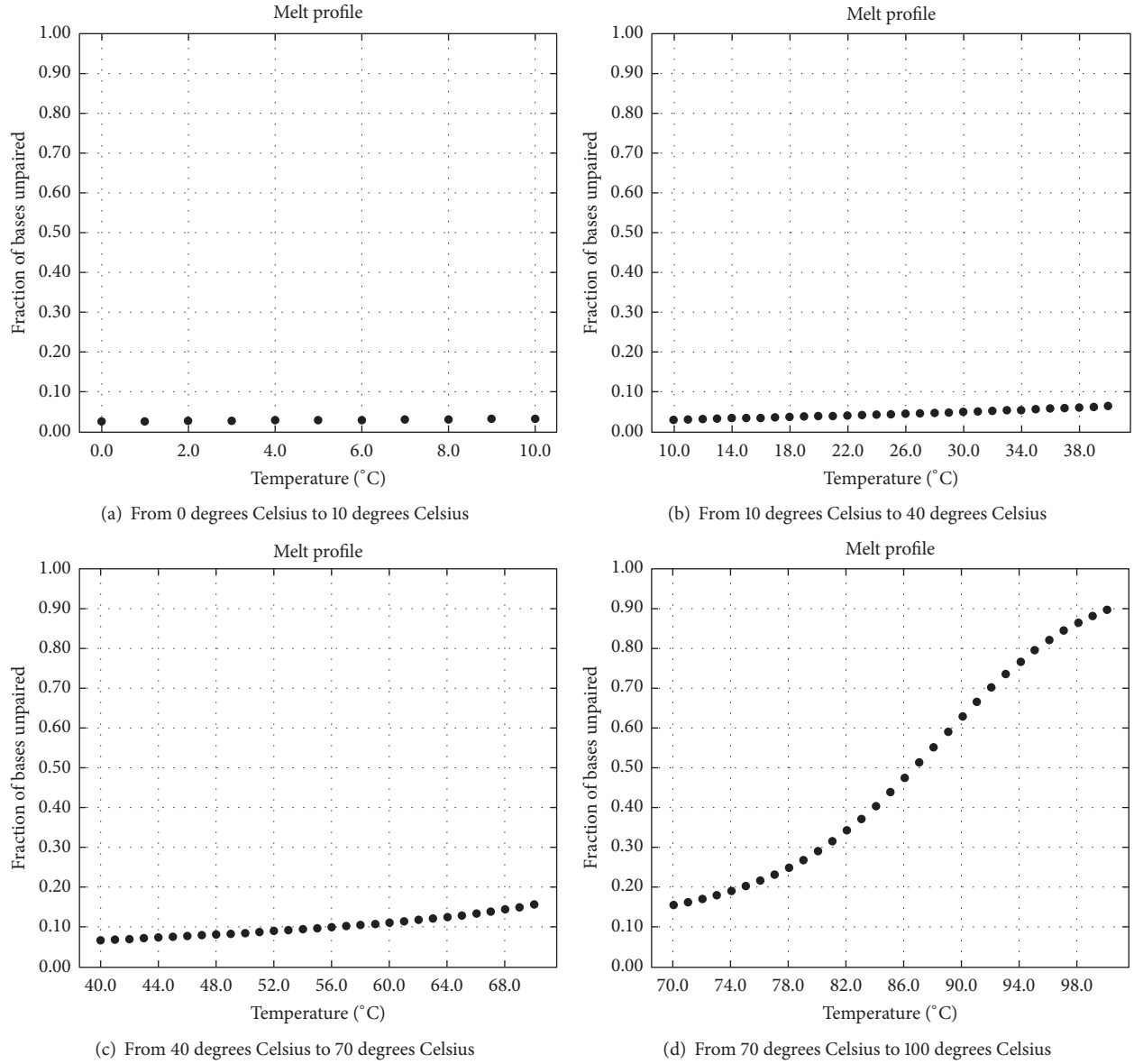
By calling the procedure for checking φ_4, Algorithm 5 can get the model checking results on this basic PTL formula. According to the experimental processes and results in Section 4.3, we can safely say that Algorithm 5, which can be employed to check the basic PTL formulas, has been effectively implemented in molecular biology.

4.4. The Effect of Reaction Temperature on the Above Experimental Results. As mentioned above, (1) the complete double strands shown in Figures 13(a) and 13(b) are formed in the process of checking φ_1; (2) the complete double strands shown in Figure 18 are formed in the process of checking φ_2; (3) the complete double strands shown in Figure 19 are formed in the process of checking φ_3. For these complete double strands which come from the hybridization, the relationships between the rates of unpaired bases and the reaction temperatures are illustrated in Figures 24, 25, and 26.

The temperatures are illustrated in Figures 24, 25, and 26. As shown in these graphs, the lower the reaction temperature, the higher the ratio of base pairing. This result suggests that the cooling target temperature (i.e., reaction temperature) has an important influence on the experimental results, and 10°C is a suitable temperature to ensure the specificity of hybridization. In comparison, the initial temperature and the cooling speed are not crucial. As far as sticker automata are concerned, one can place directly a container with some molecular reactants at room temperature, in order to obtain his/her products of hybridization. This is the standard experimental way given in [56] for sticker automata.

4.5. The Simulated Experiments on Molecular Kinetics. Regarding the complete double strands shown in Figures 13(a), 13(b), 18, and 19, the base pairings are illustrated in the corresponding graphs. This is a result of the competitive hybridization among the different kinds of molecules. In order to better observe the process of molecular competition, we design a number of experiments. With the DIZZY tool for the DNA molecular kinetics [59], the famous chemical kinetics algorithm, called Gibson-Bruck [56], is applied to compute the dynamic changes of the numbers of the various kinds of molecules in the process of hybridization. The results are illustrated in Figures 27 and 28.

Figure 27 shows the variation of the numbers of the different kinds of molecules in the process of formation of the complete double strands shown in Figures 13(a) and 13(b). The blue line in Figure 27(a) shows a change in the number of the complete double strands. Throughout the process, the number of the complete double-stranded molecules increases exponentially with time elapse, and the number of the molecular reactants decreases exponentially with time elapse, approaching zero, as shown in Figure 27(a). Within 10 seconds, the blue line begins to



(a) From 0 degrees Celsius to 10 degrees Celsius

(b) From 10 degrees Celsius to 40 degrees Celsius

(c) From 40 degrees Celsius to 70 degrees Celsius

(d) From 70 degrees Celsius to 100 degrees Celsius

Figure 24: Effect of reaction temperature on hybridization for φ_1: forming complete double strands (rate of unpaired bases).

approximate the upper bound of 60000 and it reaches the maximum value of 59405 (more than 99% of the upper bound) in 727 seconds (about 12 minutes), as shown in Figure 27(d).

The changes for the numbers of the nonspecific molecular products are illustrated in Figures 27(b) and 27(c). At first, they increase rapidly, suggesting that a large number of nonspecific products occur. In the end, they decrease exponentially, suggesting that the specific molecular products dominate the competition eventually. It can be explained that the specific molecular products have more advantages on the physical structure and thermodynamic properties than the nonspecific molecular products.

Figure 28 shows the variation of the numbers of the different kinds of molecules in the process of formation of the following two kinds of complete double strands: the one shown in Figure 18 and the one in Figure 19. These phenomena, rules, and causes in molecular kinetics are similar to the ones in Figure 27.

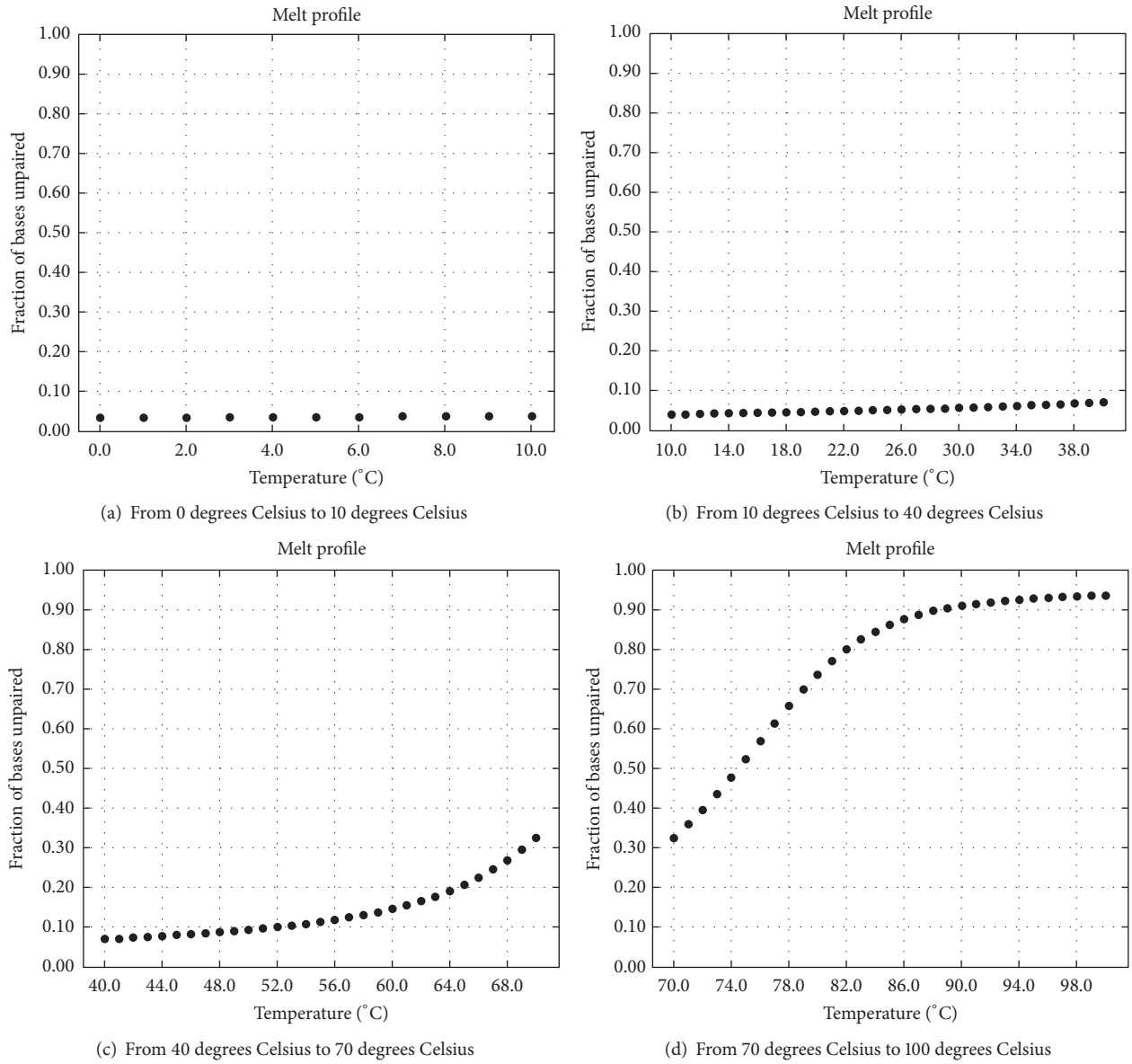
4.6. Some Comparisons among the New Method and the Related Ones

4.6.1. Comparing the New Methods with Other Related DNA-Based Ones. Table 21 gives a comparison of power between the new method and the existing DNA-based ones. From this table, the following observations can be drawn:

(i) The DNA-computing-based approach for checking the basic CTL formula EFp [53] cannot deal with any other CTL formula (including the basic formula). In comparison, the new method can conduct model checking for all of the eight basic CTL formulas via DNA molecules. In addition, the method in [53]

Table 21: A comparison of power among the various DNA model checking methods (can the method conduct DNA model checking for a given formula?).

Logic	Basic formula	Method in [53]	Method in [55]	Method in [54]	The new method	What the new method can do
LTL	pUq	No	Yes	Yes	No	—
	Fp	No	Yes	The method can be used to check. However, it is not practical to check due to the limitation of the code	No	—
	Gp	No	Yes	The method can be used to check. However, it is not practical to check due to the limitation of the code	No	—
	Xp	No	Yes	No	No	—
CTL	ApUq	No	No	No	Yes	A combination of Algorithm 1 and the experiments in [55]
	AFp	No	No	No	Yes	
	AGp	No	No	No	Yes	
	AXp	No	No	No	Yes	
	EpUq	No	No	No	Yes	The experiments for φ_1 in Section 4.1
	EFp	Yes	No	No	Yes	A combination of Algorithm 2 and the experiments in [55]
	EGp	No	No	No	Yes	
	EXp	No	No	No	Yes	
ITL	$(p_1Uq_1);(p_2Uq_2)$	No	No	No	Yes	The experiments for φ_2 in Section 4.2
	$(pUq)^*$	No	No	No	Yes	The experiments for φ_3 in Section 4.2
PTL	$((p_1Uq_1),(p_2Uq_2))$ prj $(p_3 \wedge Xq_3)$	No	No	No	Yes	The experiments for φ_4 in Section 4.3



(a) From 0 degrees Celsius to 10 degrees Celsius

(b) From 10 degrees Celsius to 40 degrees Celsius

(c) From 40 degrees Celsius to 70 degrees Celsius

(d) From 70 degrees Celsius to 100 degrees Celsius

Figure 25: Effect of reaction temperature on hybridization for φ_2: forming complete double strands (rate of unpaired bases).

cannot deal with any ITL/PTL formulas, whereas the new method can deal with them.

(ii) There are some previous DNA-computing-based approaches for checking all of the four basic LTL formulas and some popular LTL formulas [54, 55]. However, these methods cannot work on any of the CTL formula, ITL formula, and PTL formula. In comparison, the new method can conduct model checking for all of the basic formulas of the above three temporal logic types. In particular, the relationship of the expressive abilities of these three temporal logic types is shown in Figure 29.

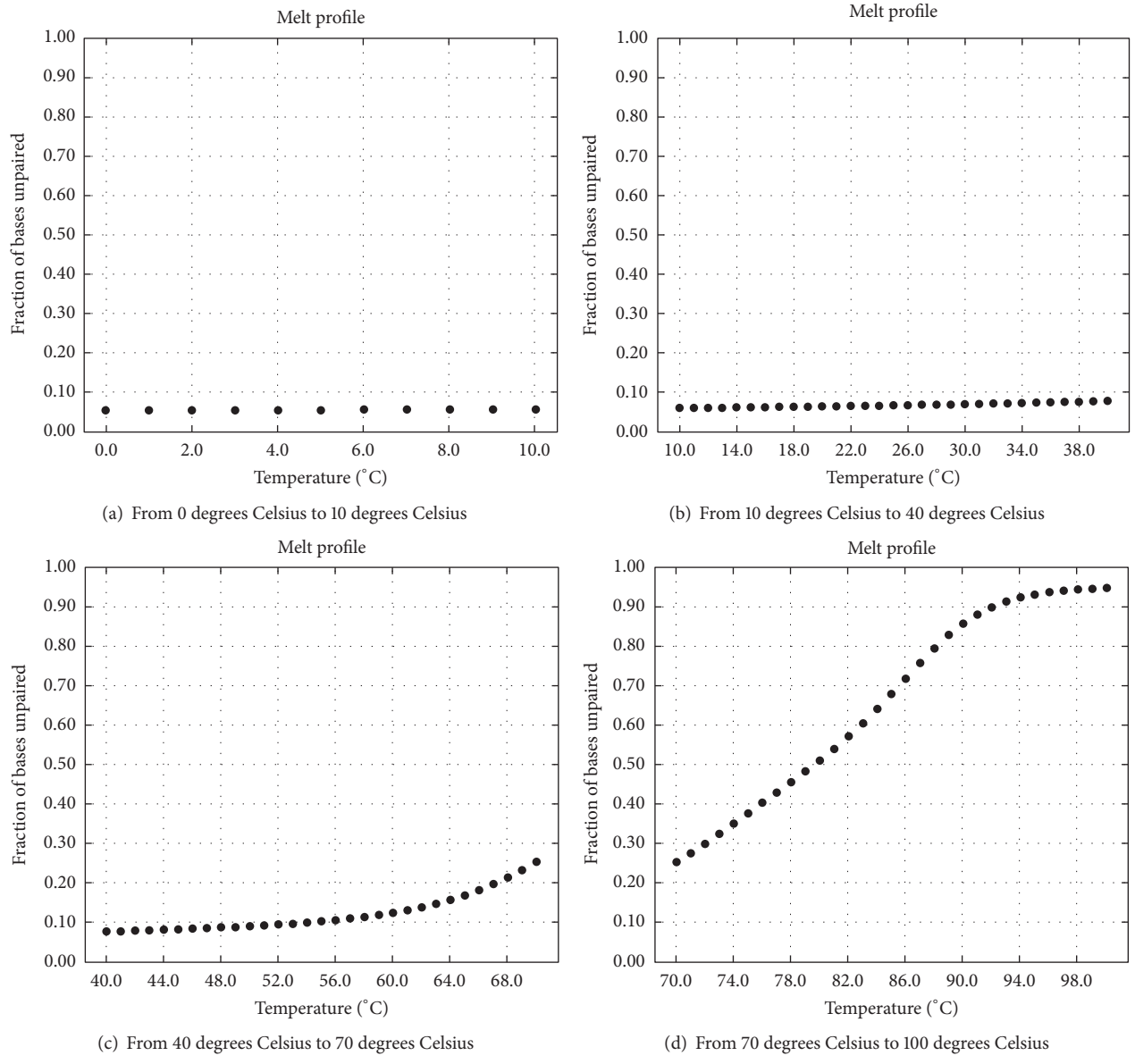
In summary, our new method extends the range of the DNA model checking, and some stronger temporal properties can be checked. In addition, the new method does not simply call the algorithm TL-MC-DNA in [55]. There are some key differences between the two methods.

(i) First, the new approach has employed some formal technique based on the semantic equivalent transformations before calling the algorithm TL-MC-DNA.

(ii) Second, the new approach extends the scope of input parameters of the algorithm TL-MC-DNA.

(iii) Third, we have designed a number of the new DNA encoding schemes which are more effective and used together with the new algorithms.

(iv) Fourth, the targeted problems are different. The algorithm TL-MC-DNA is used for the basic LTL model checking, whereas the new approach is used for the basic CTL model checking, the basic ITL model



(a) From 0 degrees Celsius to 10 degrees Celsius

(b) From 10 degrees Celsius to 40 degrees Celsius

(c) From 40 degrees Celsius to 70 degrees Celsius

(d) From 70 degrees Celsius to 100 degrees Celsius

Figure 26: Effect of reaction temperature on specific hybridization for φ_3: forming complete double strands (rate of unpaired bases).

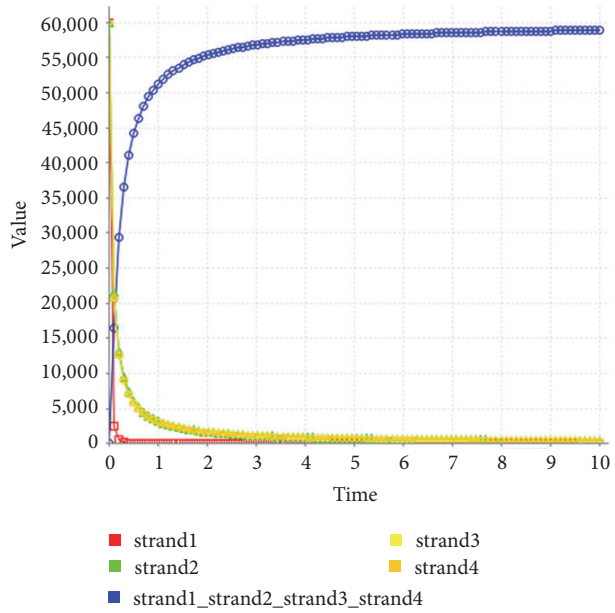
checking, and the basic PTL model checking. In other words, the latter problems are reduced to the former solved problem, using a series of logical ways and molecular biological ones. This is the research scheme in this paper.

In addition, Sections 4.1, 4.2, and 4.3 have confirmed that the simulated biochemical experiments can ensure the correctness and effectiveness for DNA model checking. In comparison, the simulated experiments on molecular kinetics in Section 4.5 further demonstrate that the DNA model checking can be biochemically implemented in acceptable time.

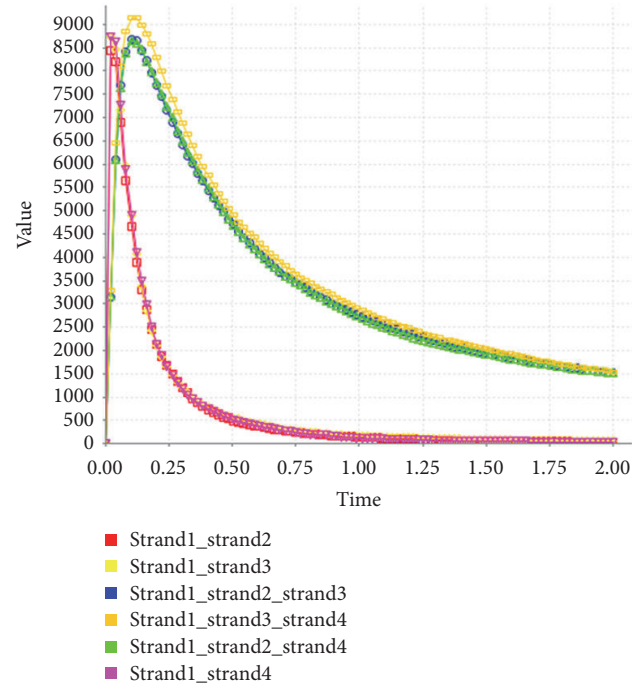
4.6.2. Comparing the New Method to the Classical Model Checking Algorithms. As shown in Section 3.1.4, the model checking method using DNA molecules is different from the model checking algorithms based on electronic computing devices, in terms of the computing mechanism due to the different computing carriers. As a result, the new method and the classical ones in [43] are complementary.

4.6.3. Additional Discussions. In [60], Professor Lamport talked about the problems that he knows with liveness. One problem is that "more than 90% (probably more than 95%) of the errors in real systems are violations of safety properties." The CTL formula AFp is usually employed to describe a safety property in practical model checking. Our new method can check AFp via DNA molecules, as illustrated in Algorithm 1. Therefore, the core of CTL in this paper is useful in practice of computing.

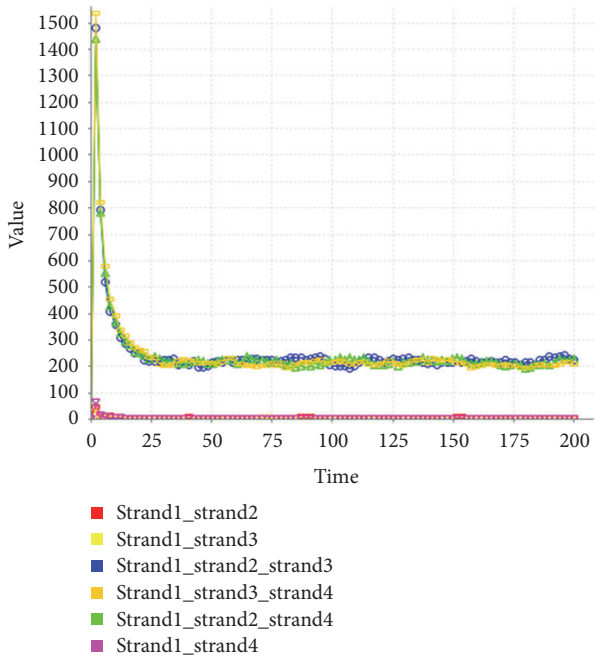
Previous research has demonstrated that the DNA model checking technique using sticker automata can be



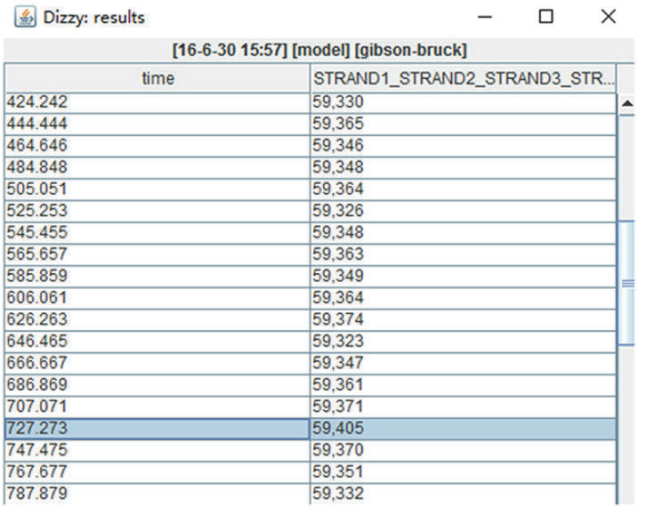
(a) Molecular number changes: reactants and specific products



(b) Molecular number changes (0–2 seconds): nonspecific products



(c) Molecular number changes (0–200 seconds): nonspecific products

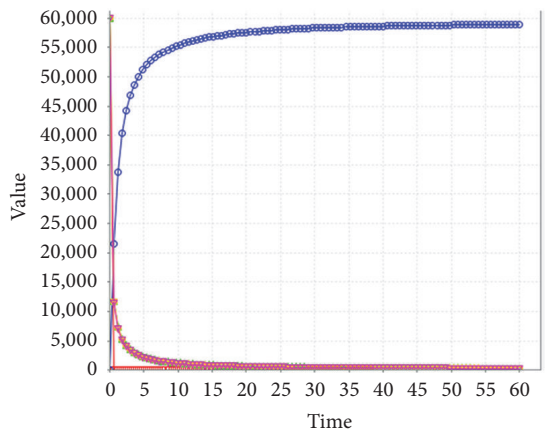


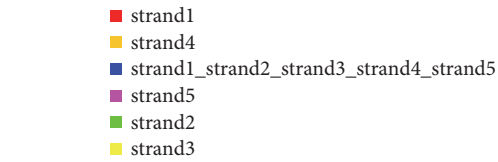
time	STRAND1_STRAND2_STRAND3_STR...
424.242	59,330
444.444	59,365
464.646	59,346
484.848	59,348
505.051	59,364
525.253	59,326
545.455	59,348
565.657	59,363
585.859	59,349
606.061	59,364
626.263	59,374
646.465	59,323
666.667	59,347
686.869	59,361
707.071	59,371
727.273	59,405
747.475	59,370
767.677	59,351
787.879	59,332

(d) Maximum molecular number of specific products

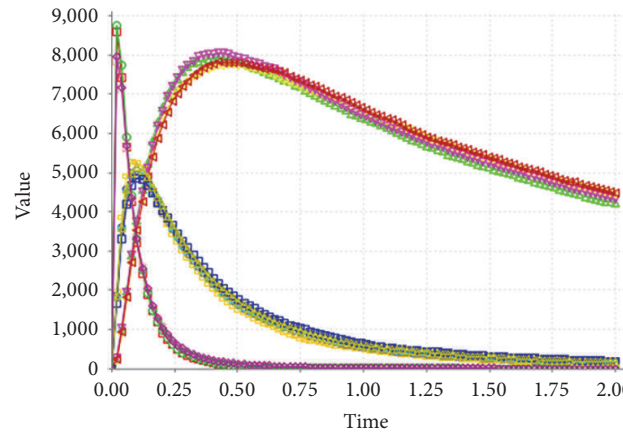
FIGURE 27: Simulated experiments in molecular kinetics: complete double strands formed for checking φ_1.

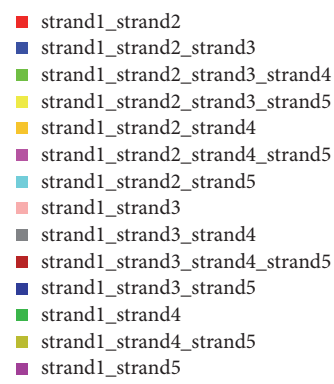
implemented in molecular biology [55], if the number of the nodes of FSA of a logical formula is not greater than 7 and the number of the edges of FSA of the logical formula is not greater than 42. It is obvious that the new method aims to deal with CTL/ITL/PTL formulas using sticker automata. Thus, our molecular biology technique mentioned above indicates that not only all of the basic formulas but also some popular CTL/ITL/PTL formulas in practice can be dealt with, by extending the new method, which is similar to the case of LTL in [55]. In addition, the methods based on sticker automata can deal with the complete decidable sets of temporal logic using some extended DNA encoding, in



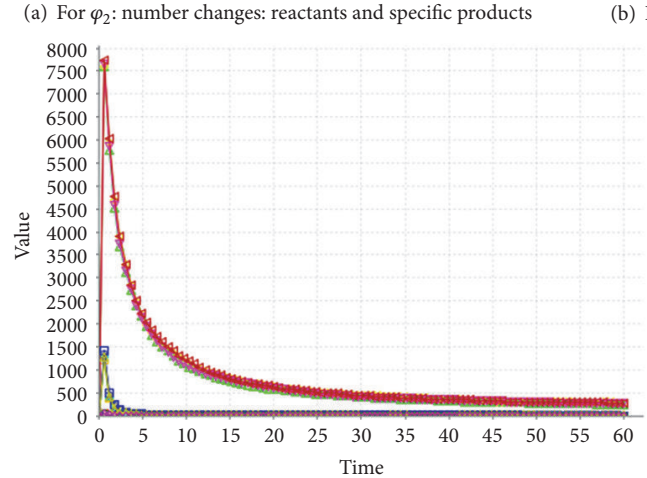


(a) For φ_2: number changes: reactants and specific products

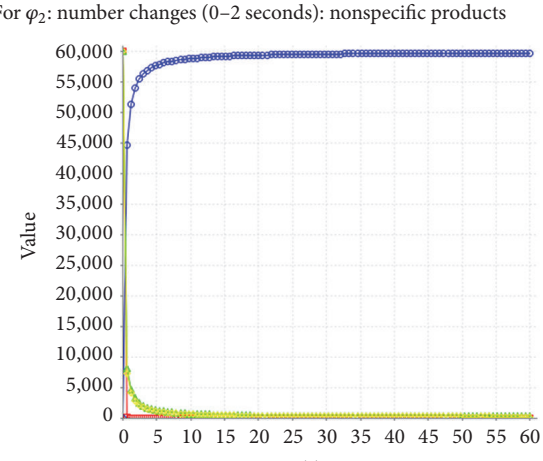




(b) For φ_2: number changes (0–2 seconds): nonspecific products



(c) For φ_2: number changes (0–60 seconds): nonspecific products



(d) For φ_3: number changes: reactants and specific products

Figure 28: Experiments in molecular kinetics: complete double strands formed for checking φ_2 and φ_3.

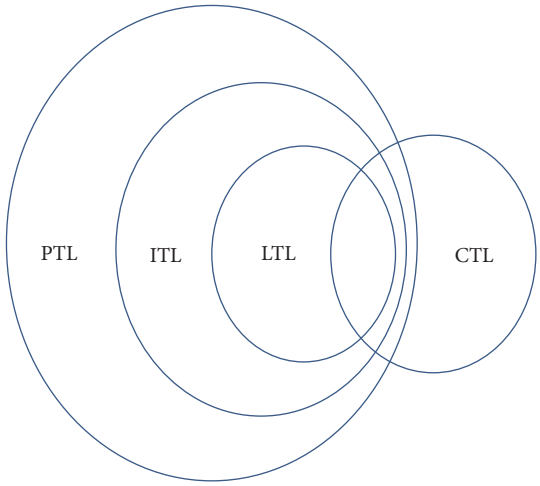


Figure 29: Comparison of power among the several logic types (comparison of action ranges among the new method and the existing ones).

theoretical computing [55]. The details of these extensions are omitted from the paper due to the scope of this paper and the limitation of space.

5. Conclusions

Early studies on DNA computing focused on nonautonomous models and algorithms. The DNA computing techniques have been optimized with self-assembly in recent years. This paper has presented a novel DNA computing method using sticker automata for model checking temporal logic formulas. Particularly, our newly developed algorithms are based on the self-assembly of sticker automata.

The state of the art of the universal DNA computer is encouraging [15, 30], and it is in a great need for new components to enhance the theoretical architecture of DNA computing, especially for the temporal logic model checking, which is a complex computational problem. The new algorithms have been implemented and model checking was conducted via DNA molecules for the basic formulas of CTL, ITL, and PTL. The model checking technique based on molecular computation has its intrinsic advantages of parallel computing, compared with the classical model checking methods. The new DNA computing approach based on sticker automata will develop a molecular solution and expand the previous DNA computational problem library.

There are several directions that can be further explored based on the new method described in this paper. One area is to apply the cellular model checking technique for genomic research. For example, it can be used to study DNA repairing and mismatching during cell division, which was believed to be associated with cancer occurrence [56]. In order to improve the ability of discovery and repair of abnormal genes not only at the structural level but also at the functional level, it is necessary to study the temporal and spatial expression of genes. Some previous research has employed the cellular computation technique to provide an autonomous intelligent method for the molecular diagnosis and treatment of some human diseases which are caused by genetic mutation [56]. However, the new method can deal with more temporal relationships. Therefore, one of the future works of our study is to incorporate our new DNA computing approach for model checking temporal logic into the artificial controlled gene repair techniques. This will develop a molecular means for the early detection, diagnosis, and treatment of cancer. It will impact the prognosis and the survival rate of cancer patients.

A specific future application of our method is to study the gastric cardiac cancer for the stage of gastric inflammation and precancerous lesions, which showed some abnormal behavioral changes in genes with temporal characteristics. To give an example, previous study discovered a susceptibility gene locus of gastric cardiac cancer in the Han population of Northern China [61]. Future research will focus on how to embed the autonomous model checking method into living human cells. Such an approach can be used to develop a molecular robot technique to repair susceptibility loci or DNA mismatches in the gastric cancer cells or the normal cells. To be a little more specific, the basic CTL formulas can be applied to describe the branch temporal relationships among dynamic changes in genes, the basic ITL formulas for the simple interval relationships among dynamic changes in genes, and the basic PTL formula for the general relationships between intervals and their effects of dynamic changes in genes. These basic temporal logical formulas are sufficient to describe the dynamic changes of genes and no other formulas are needed.

Appendix

Sections 2.1, 2.2, 2.3, and 2.4 give the intuitive meaning of the four temporal logic types. The Appendix gives the formal descriptions of these temporal logic types.

A. Linear Temporary Logic (LTL)

Definition A.1 (syntax of propositional LTL [43], LTL for short). Let AP be a set of atomic propositions; then,

(i) for all $p \in \mathrm{AP}$, p is a LTL formula;

(ii) if φ is a LTL formula, then $\neg\varphi$ is a LTL formula;

(iii) if φ and ψ are LTL formulas, then $\varphi \vee \psi$ is a LTL formula;

(iv) if φ is a LTL formula, then $\mathrm{X}\varphi$ is a LTL formula;

(v) if φ and ψ are LTL formulas, then $\varphi \mathrm{U} \psi$ is a LTL formula.

Definition A.2 (derived LTL formulas). $\mathrm{F}\varphi = true\mathrm{U}\varphi$, $\mathrm{G}\varphi = \neg\mathrm{F}\neg\varphi$, $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$.

Definition A.3 (model for LTL). A LTL model is a triple $\mathrm{M} = (S, R, LABEL)$, where

(i) S is a nonempty denumerable set of states,

(ii) $R : S \to S$ assigns to $s \in S$ its unique successor state $R(s)$,

(iii) $LABEL : S \to 2^{AP}$ assigns to each state $s \in S$ the atomic propositions $LABEL(s)$ that are valid in s.

Definition A.4 (semantic of LTL). Let $p \in$ AP be an atomic proposition, $M = (S, R, LABEL)$ a LTL model, $s \in S$, and φ, ψ PLTL formulas; the satisfaction relation $\models$ is defined by

(i) $s \models p$ if and only if (iff, for short) $p \in LABEL(s)$,

(ii) $s \models \neg\varphi$ if and only if $\neg(s \models \varphi)$,

(iii) $s \models \varphi \vee \psi$ if and only if $(s \models \varphi) \vee (s \models \psi)$,

(iv) $s \models X\varphi$ if and only if $R(s) \models \varphi$,

(v) $s \models \varphi U \psi$ if and only if $\exists j \geq 0.\ R^j(s) \models \psi \wedge (\forall 0 \leq k < j.\ R^k(s) \models \varphi)$.

Here, $R^0(s) \models s$, and $R^{n+1}(s) = R^n(R(s))$ for any $n \geq 0$, where $R(s) = s'$ and the state s' is called a direct successor of s.

B. Interval Temporal Logic (ITL)

Definition B.1 (syntax of propositional ITL [50], ITL for short). (i) For all $p \in$ AP, p is an ITL formula.

(ii) If φ, ψ are ITL formulas, so are the constructs $\neg\varphi$, $\varphi \vee \psi$, $X\varphi$, $\varphi; \psi$, φ^*.

Definition B.2. An interval of states is defined and also denoted as $\sigma = \langle s_0, s_1, \ldots, s_i, \ldots \rangle$, where s_i is a state.

Definition B.3. An interpretation is a quadruple $I = (\sigma, i, k, j)$, where σ is a sequence of states over $\langle s_i, \ldots, s_k, \ldots, s_j \rangle$, $i, k, j \in N$, and s_k is the current state. We use the notation $\text{len}(\sigma) \dashv \sigma \models j - i$ for the number of states in interval.

Definition B.4 (semantic of ITL). Let $c \in N$ and $s_p(k)$ be the true value of $p \in$ AP in state s_k. The satisfaction relation is inductively defined as follows:

(1) $I \models p$ iff $s_p(k) = \text{true}$,

(2) $I \models \neg\varphi$ iff $\neg(I \models \varphi)$,

(3) $I \models \varphi_1 \vee \varphi_2$ iff $I \models \varphi_1$ or $I \models \varphi_2$,

(4) $I \models skip$ iff $\text{len}(\sigma) = 1$,

(5) $I \models X\varphi$ iff $(\sigma, i, k+1, j) \models \varphi$,

(6) $I \models \varphi_1; \varphi_2$ iff $\exists r, k \leq r \leq j$, such that $(\sigma, i, k, r) \models \varphi_1$ and $(\sigma, i, k, r) \models \varphi_2$,

(7) $I \models \varphi^*$ iff (i) there exist finitely many $r_0, \ldots, r_n \in N_\omega$, such that $k = r_0 \leq r_1 \leq \cdots \leq r_{n-1} \leq r_n = j$, $(\sigma, i, k, r_0) \models \varphi$, and for every $1 \leq l \leq n$, $(\sigma, r_{l-1}, r_{l-1}, r_l) \models \varphi$; (ii) or $k = j$.

Definition B.5 (derived ITL formulas). $F\varphi = true; \varphi$, $G\varphi = \neg F \neg\varphi$, $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$, $\varphi_1 \parallel \varphi_2 = \varphi_1 \wedge (\varphi_2; true) \vee \varphi_2 \wedge (\varphi_1; true)$.

C. Computational Tree Logic (CTL)

Definition C.1 (syntax of propositional CTL [49], CTL for short). Given a set of basic propositions P, a CTL formula ϕ on P is any recursive combination of the propositions of P through the Boolean connectives *not* ($\neg\phi$), *and* ($\phi_1 \wedge \phi_2$), and through the temporal operators *existential until* ($E[\phi_1 U \phi_2]$) *universal until* ($A[\phi_1 U \phi_2]$):

$$\phi = p \mid \neg\phi_1 \mid \phi_1 \wedge \phi_2 \mid E[\phi_1 U \phi_2] \mid A[\phi_1 U \phi_2] \tag{C.1}$$

with $p \in P$.

Definition C.2 (semantic of CTL). Given a state transition system S and a state $s_0 \in S$, a CTL formula ϕ is interpreted on S according to the satisfaction relation $s_0 \models \phi$ (read s_0 satisfies ϕ) defined by the following clauses:

(i) $s_0 \models p$ iff the proposition p holds in s_0;

(ii) $s_0 \models \neg\phi_1$ iff ϕ_1 is not satisfied in s_0;

(iii) $s_0 \models \phi_1 \wedge \phi_2$ iff both ϕ_1 and ϕ_2 are satisfied in s_0;

(iv) $s_0 \models E[\phi_1 U \phi_2]$ iff there exists a path starting from s_0 which reaches a future state in which ϕ_2 is satisfied and such that ϕ_1 is satisfied in every intermediate state;

(v) $s_0 \models A[\phi_1 U \phi_2]$ iff along every path starting from s_0 a future state is eventually reached in which ϕ_2 is satisfied, and ϕ_1 is satisfied in all the intermediate states.

Definition C.3 (short hands). Further operators can be defined as short hands by combining Boolean connectives and temporal operators.

(i) Boolean connectives such as *or* ($\vee$) and *implies* ($\to$), with their usual meaning, can be derived as

$$\begin{aligned} \phi_1 \vee \phi_2 &= \neg(\neg\phi_1 \wedge \neg\phi_2), \\ \phi_1 &\longrightarrow \phi_2 = \neg(\phi_1 \wedge (\neg\phi_2)). \end{aligned} \tag{C.2}$$

(ii) Temporal operators *eventually* (F) and *always* (G), with existential and universal path quantification, are derived as follows:

$$\begin{aligned} EF\phi_2 &== E[true U \phi_2], \\ AF\phi_2 &== A[true U \phi_2], \\ EG\phi_1 &== \neg AF(\neg\phi_1), \\ AG\phi_1 &== \neg EF(\neg\phi_1). \end{aligned} \tag{C.3}$$

D. Projection Temporal Logic (PTL)

Definition D.1 (syntax of propositional PTL [52], PTL for short). Let p and q be atomic propositions. The formula P of PPTL is given by the following grammar:

$$P ::= p \mid XP \mid \neg P \mid P_1 \vee P_2 \mid (P_1, \ldots, P_m)\ prj\ P, \tag{D.1}$$

where $p \in Prop, P_1, \ldots, P_m$ and P are all well-formed PPTL formulas.

Definition D.2 (semantic of PTL).

(1) States. Following the definition of Kripke's structure, we define a state s over $Prop$ to be a mapping from $Prop$ to $B = \{true, false\}$, $s : Prop \to B$. We will use $s[p]$ to denote the valuation of p at state s.

(2) Intervals. An interval σ is a nonempty sequence of states, which can be finite or infinite. The length, $|\sigma|$, of σ is ω if σ is infinite, with the number of states minus 1 if σ is finite. To have a uniform notation for both finite and infinite intervals, we will use extended integers as indices. That is, we consider the set N_0 of nonnegative integers and ω, $N_\omega = N_0 \cup \{\omega\}$, and extend the comparison operators, $=$, $<$, $\leq$, to N_ω by considering $\omega = \omega$, and for all $i \in N_0$, $i < \omega$. Moreover, we define $\preceq$ as $\leq - \{(\omega, \omega)\}$. To simplify definitions, we will denote σ by $\langle s_0, \ldots, s_{|\sigma|}\rangle$, where $s_{|\sigma|}$ is undefined if σ is infinite. With such a notation, $\sigma_{(i,\ldots,j)}$ $0 \leq i \preceq j \leq |\sigma|$ denotes the subinterval $\langle s_i, \ldots, s_j\rangle$ and $\sigma^{(k)}$ $(0 \leq k \preceq |\sigma|)$ denotes $\langle s_k, \ldots, s_{|\sigma|}\rangle$. The concatenation of a finite σ with another interval (or empty string) σ' is denoted by $\sigma \cdot \sigma'$.

Let $\sigma = \langle s_0, s_1, \ldots, s_{|\sigma|}\rangle$ be an interval and $r_1, \ldots, r_h$ be integers $(h \geq 1)$ such that $0 \leq r_1 \leq r_2 \leq \cdots \leq r_h \preceq |\sigma|$. The projection of σ onto $r_1, \ldots, r_h$ is the interval (namely, projected interval)

$$\sigma \downarrow (r_1, \ldots, r_h) = \langle s_{t1}, s_{t2}, \ldots, s_{tl}\rangle, \tag{D.2}$$

where $t_1, \ldots, t_l$ is obtained from $r_1, \ldots, r_h$ by deleting all duplicates. That is, $t_1, \ldots, t_l$ is the longest strictly increasing subsequence of $r_1, \ldots, r_h$. For instance,

$$\langle s_0, s_1, s_2, s_3, s_4\rangle \downarrow (0, 0, 2, 2, 2, 3) = \langle s_0, s_2, s_3\rangle. \tag{D.3}$$

This is convenient to define an interval obtained by taking the endpoints (rendezvous points) of the intervals over which $P_1, \ldots, P_m$ are interpreted in the projection construct.

(3) Interpretations and Satisfaction Relation. An interpretation is a triple $I = (\sigma, k, j)$, where σ is an interval, k is an integer, and j is an integer or ω such that $k \preceq j \leq |\sigma|$. We use the notation $(\sigma, k, j) \models P$ to denote that formula P is interpreted and satisfied over the subinterval $\langle s_k, \ldots, s_j\rangle$ of σ with the current state being s_k.

The satisfaction relation $(\models)$ is inductively defined as follows:

(1) $I \models p$ iff $s_k[p] = true$, for any given proposition p;

(2) $I \models \neg P$ iff $I \not\models P$;

(3) $I \models P \vee Q$ iff $I \models P$ or $I \models Q$;

(4) $I \models \mathrm{X}P$ iff $k < j$ and $(\sigma, k+1, j) \models P$;

(5) $I \models (P_1, \ldots, P_m)\ prj\ Q$ if there exist integers $k = r_0 \leq r_1 \leq \cdots \leq r_m \leq j$ such that $(\sigma, r_0, r_1) \models P_1$, $(\sigma, r_{l-1}, r_l) \models P_l$, $1 < l \leq m$, and $(\sigma', 0, |\sigma'|) \models Q$ for one of the following σ':

(a) $r_m < j$ and $\sigma' = \sigma \downarrow (r_0, \ldots, r_m) \cdot \sigma_{(r_m+1\cdots j)}$;

(b) $r_m = j$ and $\sigma' = \sigma \downarrow (r_0, \ldots, r_h)$ for some $0 \leq h \leq m$.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China under Grants U1204608 and 61572444, China Postdoctoral Science Foundation under Grant 2015M572120, and Science Foundation for the Excellent Young Teachers in Henan Province under Grant 2014GGJS-001, as well as Scientific and Technological Project of Henan Province under Grants 152300410055 and 152102410033.

References

[1] L. M. Adleman, "Molecular computation of solutions to combinatorial problems," *Science*, vol. 266, no. 5187, pp. 1021–1023, 1994.

[2] R. J. Lipton, "DNA solution of hard computational problems," *Science*, vol. 268, no. 5210, pp. 542–545, 1995.

[3] Q. Ouyang, P. D. Kaplan, S. Liu, and A. Libchaber, "DNA solution of the maximal clique problem," *Science*, vol. 17, pp. 446–449, 1997.

[4] Y. Benenson, T. Paz-Elizur, R. Adar, E. Keinan, Z. Llvneh, and E. Shapiro, "Programmable and autonomous computing machine made of biomolecules," *Nature*, vol. 414, no. 6862, pp. 430–434, 2001.

[5] L. Adleman, "On Constructing A Molecular Computer," in *Proceedings of the conference on DNA based Computers*, Princeton University, Princeton, New Jersey, USA, 1995.

[6] S. Roweis, E. Winfree, R. Burgoyne et al., "A sticker-based model for DNA computation," *Journal of Computational Biology*, vol. 5, no. 4, pp. 615–629, 1998.

[7] J. Kuramochi and Y. Sakakibara, "Intensive in vitro experiments of implementing and executing finite automata in test tube," in *Proceedings of the International Workshop on DNA-Based Computers, DNA Computing*, vol. 3892 of *Lecture Note in Computer Science*, pp. 193–202, Springer press, Berlin, Heidelberg, Germany.

[8] I. M. Martínez-Pérez, G. Zhang, Z. Ignatova, and K.-H. Zimmermann, "Computational genes: A tool for molecular diagnosis and therapy of aberrant mutational phenotype," *BMC Bioinformatics*, vol. 8, no. 1, 365 pages, 2007.

[9] P. Yin, A. J. Turberfield, S. Sahu, and J. H. Reif, "Design of an autonomous DNA nanomechanical device capable of universal computation and universal translational motion," in *DNA Computing*, vol. 3384 of *Lecture Note in Computer Science*, pp. 426–444, Springer, Milan, Italy, 2005, Tenth International Meeting on DNA Computing.

[10] J. Xu, X. Qiang, Y. Yang et al., "An unenumerative DNA computing model for vertex coloring problem," *IEEE Transactions on Nanobioscience*, vol. 10, no. 2, pp. 94–98, 2011.

[11] J.-H. Xiao and J. Xu, "The DNA computation model based on giant magnetoresistance for SAT problem," *Chinese Journal of Computers*, vol. 36, no. 4, pp. 829–835, 2013 (Chinese).

[12] C. Zhang, L.-N. Ma, Y.-F. Dong et al., "Molecular logic computing model based on DNA self-assembly strand branch

migration," *Chinese Science Bulletin*, vol. 57, no. 31, pp. 2909–2915, 2012.

[13] C. Zhang, J. Yang, and J. Xu, "Molecular logic computing model based on self-assembly of DNA nanoparticles," *Chinese Science Bulletin*, vol. 56, no. 33, pp. 3566–3571, 2011.

[14] K.-L. Li, X. Luo, F. Wu et al., "An algorithm in tile assembly model for maximum clique problem," *Journal of Computer Research and Development*, vol. 50, no. 3, pp. 666–675, 2013 (Chinese).

[15] J. Xu, "Probe machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 7, pp. 1405–1416, 2016.

[16] J. Xu, X. Qiang, K. Zhang et al., "A parallel type of DNA computing model for graph vertex coloring problem," in *Proceedings of the 2010 IEEE 5th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010*, pp. 231–235, IEEE, Changsha, China, September 2010.

[17] F. Wu and K.-L. Li, "An algorithm in tile assembly model for N queen problem," *Tien Tzu Hsueh Pao/Acta Electronica Sinica*, vol. 41, no. 11, pp. 2174–2180, 2013 (Chinese).

[18] X. Zhou, K.-L. Li, G.-X. Le et al., "A volume molecular solution for the maximum matching problem on DNA-based computing," *Journal of Computer Research and Development*, vol. 48, no. 11, pp. 2147–2154, 2011 (Chinese).

[19] F. Wu, K. Li, A. Sallam, and X. Zhou, "A molecular solution for minimum vertex cover problem in tile assembly model," *Journal of Supercomputing*, vol. 66, no. 1, pp. 148–169, 2013.

[20] X. Zhou, Y. Zhou, K. Li, A. Sallam, and K. Li, "Molecular solutions for minimum and exact cover problems in the tile assembly model," *Journal of Supercomputing*, vol. 69, no. 2, pp. 976–1005, 2014.

[21] K.-L. Li, F.-J. Yao, J. Xu et al., "An O(1.414n) volume molecular solutions for the subset-sum problem on dna-based supercomputing," *Chinese Journal of Computers*, vol. 30, no. 11, pp. 1947–1953, 2007 (Chinese).

[22] X. Zhou, K. Li, M. Goodman et al., "A novel approach for the classical ramsey number problem on DNA-based supercomputing," *Communications in Mathematical and in Computer Chemistry*, vol. 66, no. 1, pp. 347–370, 2011.

[23] X. Liu, L. Xiang, and X. Wang, "Spatial cluster analysis by the adleman-lipton DNA computing model and flexible grids," *Discrete Dynamics in Nature and Society*, vol. 2012, Article ID 894207, 32 pages, 2012.

[24] X. Liu and J. Xue, "Spatial cluster analysis by the bin-packing problem and DNA computing technique," *Discrete Dynamics in Nature and Society*, vol. 2013, Article ID 891428, 8 pages, 2013.

[25] H. Taghipour, M. Rezaei, and H. A. Esmaili, "Solving the 0/1 knapsack problem by a biomolecular DNA computer," *Advances in Bioinformatics*, vol. 2013, Article ID 341419, 6 pages, 2013.

[26] J. Yang, C. Dong, Y. Dong et al., "Logic nanoparticle beacon triggered by the binding induced effect of multiple inputs," *ACS Applied Materials & Interfaces*, vol. 6, no. 16, pp. 14486–14492, 2014.

[27] C. Zhang, L. Wu, J. Yang et al., "A molecular logical switch beacon controlled by thiolated DNA signals," *Chemical Communications*, vol. 49, no. 96, pp. 11308–11310, 2013.

[28] C. Zhang, J. Ma, and J. Yang, "Nanoparticle aggregation logic computing controlled by DNA branch migration," *Applied Physics Letters*, vol. 103, no. 9, pp. 93–106, 2013.

[29] M. Chen, X.-Q. Chen, L. Zhang, and J. Xu, "A biobrick inversion cellular computing model for satisfiability problem," *Jisuanji Xuebao/Chinese Journal of Computers*, vol. 36, no. 12, pp. 2537–2544, 2013.

[30] J. Xu, "Forthcoming era of biological computer," *Bulletin of the Chinese Academy of Sciences*, vol. 29, no. 1, pp. 42–54, 2014 (Chinese).

[31] M. Karakose and U. Cigdem, "QPso-based adaptive DNA computing algorithm," *The Scientific World Journal*, vol. 2013, Article ID 160687, 8 pages, 2013.

[32] X. Zheng, B. Wang, C. Zhou, X. Wei, and Q. Zhang, "Parallel DNA arithmetic operation with one error detection based on 3-moduli set," *IEEE Transactions on Nanobioscience*, vol. 15, no. 5, pp. 499–507, 2016.

[33] P. Y. De Silva and G. U. Ganegoda, "New Trends of Digital Data Storage in DNA," *BioMed Research International*, vol. 2016, Article ID 8072463, 2016.

[34] W.-L. Chang and A. V. Vasilakos, "DNA algorithms of implementing biomolecular databases on a biological computer," *IEEE transactions on nanobioscience*, vol. 14, no. 1, pp. 104–111, 2014.

[35] A. Eshra and A. El-Sayed, "An odd parity checker prototype using DNAzyme finite state machine," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 2, pp. 316–324, 2014.

[36] J. Wang and R. Huang, "Design and implementation of a microfluidic half adder chip based on double-stranded DNA," *IEEE Transactions on Nanobioscience*, vol. 13, no. 2, pp. 146–151, 2014.

[37] S. Zhou, B. Wang, X. Zheng, and C. Zhou, "An image encryption scheme based on DNA computing and cellular automata," *Discrete Dynamics in Nature and Society*, vol. 2016, Article ID 5408529, 9 pages, 2016.

[38] B. Wang, Y. Xie, S. Zhou, C. Zhou, and X. Zheng, "Reversible Data Hiding Based on DNA Computing," *Computational Intelligence and Neuroscience*, vol. 2017, Article ID 7276084, 9 pages, 2017.

[39] H.-C. Huang, S. S.-D. Xu, and H.-S. Hsu, "Hybrid taguchi DNA swarm intelligence for optimal inverse kinematics redundancy resolution of six-DOF humanoid robot arms," *Mathematical Problems in Engineering*, vol. 2014, Article ID 358269, 9 pages, 2014.

[40] M. Baygin and M. Karakose, "Immunity-based optimal estimation approach for a new real time group elevator dynamic control application for energy and time saving," *The Scientific World Journal*, vol. 2013, Article ID 805343, 12 pages, 2013.

[41] H. Jiao, Y. Zhong, and L. Zhang, "Artificial DNA computing-based spectral encoding and matching algorithm for hyperspectral remote sensing data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 10, pp. 4085–4104, 2012.

[42] H. Jiao, Y. Zhong, and L. Zhang, "An unsupervised spectral matching classifier based on artificial DNA computing for hyperspectral remote sensing imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 8, pp. 4524–4538, 2014.

[43] E. Clarke et al., *Model Checking*, MIT press, Massachusetts, USA, 1999.

[44] J. Barnat, P. Bauch, L. Brim et al., "Designing fast LTL model checking algorithms for many-core GPUs," *Journal of Parallel and Distributed Computing*, vol. 72, no. 9, pp. 1083–1097, 2012.

[45] R. Carbone, "LTL model-checking for security protocols," *AI Communications*, vol. 24, no. 4, pp. 281–283, 2011.

[46] A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay, and J.-F. Raskin, "Featured transition systems: Foundations for verifying variability-intensive systems and their application to

LTL model checking," *IEEE Transactions on Software Engineering*, vol. 39, no. 8, pp. 1069–1089, 2013.

[47] A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pp. 46–57, IEEE Computer Society, Washington, DC, USA, 1977.

[48] M. Ben-Ari, A. Pnueli, and Z. Manna, "The temporal logic of branching time," *Acta Informatica*, vol. 20, no. 3, pp. 207–226, 1983.

[49] E. A. Emerson and E. M. Clarke, "Using branching time temporal logic to synthesize synchronization skeletons," *Science of Computer Programming*, vol. 2, no. 3, pp. 241–266, 1982.

[50] B. Moszkowski, *Reasoning about Digital Circuits [PhD thesis]*, Department of Computer Science, Stanford University, 1983.

[51] Z. Chaochen, C. A. R. Hoare, and A. P. Ravn, "A calculus of durations," *Information Processing Letters*, vol. 40, no. 5, pp. 269–276, 1991.

[52] Z. Duan, C. Tian, and L. Zhang, "A decision procedure for propositional projection temporal logic with infinite models," *Acta Informatica*, vol. 45, no. 1, pp. 43–78, 2008.

[53] E. A. Emerson, K. D. Hager, and J. H. Konieczka, "Molecular model checking," *International Journal of Foundations of Computer Science*, vol. 17, no. 4, pp. 733–741, 2006.

[54] W.-J. Zhu, Q.-L. Zhou, and Y.-L. Li, "LTL model checking based on DNA computing," *Acta Electronica Sinica*, vol. 44, no. 6, pp. 1265–1271, 2016 (Chinese).

[55] W.-J. Zhu, Q.-L. Zhou, and Q.-X. Zhang, "A LTL model checking approach based on DNA computing," *Chinese Journal of Computers*, vol. 39, no. 12, pp. 2578–2597, 2016 (Chinese).

[56] K. Zimmermann, Z. Ignatova, and I. Martinez-Pérez, *DNA Computing Models*, Springer press, New York, NY, USA, 2008.

[57] NUPACK, 2015, http://www.nupack.org/partition/new.

[58] Y.-F. Wang and G.-Z. Cui, *The design and optimization of DNA coding sequence*, Publishing House of Electronics Industry, Beijing, China, 2013.

[59] Dizzy Home Page, 2008, http://magnet.systemsbiology.net/software/Dizzy/.

[60] "Liveness Manifestos," 2016, http://www.cs.nyu.edu/acsys/beyond-safety/liveness.htm.

[61] L. D. Wang, F. Y. Zhou, X. M. Li et al., "Genome-wide association study of esophageal squamous cell carcinoma in Chinese subjects identifies susceptibility loci at PLCE1 and C20orf54," *Nature Genetics*, vol. 42, pp. 759–763, 2010.