Autonomous forward inference via DNA computing

Fu Yan[1], Li Gen[2], Li Yin[1] and Meng Dazhi[1] *

(1. DNA Computing Research Group, Department of Operational Research and Cybernetics, Institute of Applied Science, Beijing University of Technology, Beijing 100022, China; 2. Department of Informatics and Mathematical Modeling, Technical University of Denmark, Copenhagen, Denmark)

Accepted on November 30, 2006

Abstract Recent studies direct the researchers into building DNA computing machines with intelligence, which is measured by three main points: autonomous, programmable and able to learn and adapt. Logical inference plays an important role in programmable information processing or computing. Here we present a new method to perform autonomous molecular forward inference for expert system. A novel repetitive recognition site (RRS) technique is invented to design rule-molecules in knowledge base. The inference engine runs autonomously by digesting the rule-molecule, using a Class IIB restriction enzyme PpiI. Concentration model has been built to show the feasibility of the inference process under ideal chemical reaction conditions. Moreover, we extend to implement a triggering communication between molecular automata, as a further application of the RRS technique in our model.

Keywords: RRS technique, forward inference, triggering communication, molecular automaton.

In the past few years in DNA computing, studies[1-12] have provided us a notion about the programmable and interactive attribute of DNA computing machine. The basic framework of DNA computing machine contains three main parts: sensor, processor, and actuator[13]. Sensors collect variant information from a large environment, and actuators affect the environment based on the decision made by the processor. The processor is the foremost core, because its inner program or algorithm determines the effectiveness of DNA computing machine. To implement programmable and interactive automata to perform complex tasks, a self-controlled mechanism, even with the ability of adaptation to the changing environment, is necessary. Therefore, how to design an autonomous processor is in concern. In this paper, we present a method for autonomous forward inference, which may be useful in building the processor of a DNA computing machine. For a direct application, our model implements a large and parallel forward inference mechanism for expert system. A series of early studies[14-16] have shown how to build forward and backward inference chaining for expert system. In our model, a rule in the knowledge base is able to match more than one conclusion of the other fired rules, which overcomes the restriction of rules confronted to early work[15].

In this model, we introduce a Class IIB restriction enzyme PpiI (5′-NNNNN ^ NNNNNNN GAAC NNNNN CTC NNNNNNNN ^ NNNNN-3′)[17] to run the forward inference. Input facts and rules in the knowledge base of an expert system are encoded in dsDNA with sticky ends. Each fact-molecule contains the left moiety of PpiI recognition site (RS for short): GAAC, while each rule-molecule contains several right moiety of RS: CTC. We nominate this scheme as RRS technique, i.e. repetitive recognition site technique. Therefore, if some fact is present, its sticky end will hybridize to the corresponding rule-molecule to form an integral RS (…GAAC NNNNN CTC…). If all the facts for some rules are present, enzyme PpiI will digest the rule-molecule all the way till it releases its conclusion part. And then the conclusion may participate into matching course as a newly produced input fact. The whole course of inference is autonomous.

In the later section, we show that the RRS technique can be further extended to implement a triggering communication between molecular automata. Beyer et al. have done related work on molecular translator, for the purpose of linking several computational devices together to perform more complex tasks. Here we demonstrate that our model can be another molecular translator, performing a complex triggering communication between different automata.

* To whom correspondence should be addressed. E-mail: dzhmeng@bjut.edu.cn

1 Brief introduction to the inference mechanisms of expert system

Knowledge base is the basis of an expert system, comprised of rules, individual facts and complex objects. To simplify the problem, we reduce the knowledge base to sets of rules and facts only. The mechanism of inference is defined as a way in solving a problem based on logic. Generally, there are three main categories of inference mechanisms: the forward inference, the backward inference and the one mixed in between. The forward inference is known as an inference path from facts to conclusions based on certain rules. An IF-THEN rule may contain several premises and one conclusion. In condition that there are all the facts corresponding to the premises contained in a rule, the rule is fired and makes its conclusion. This conclusion may act as another input fact to match other rules, if it is not a final conclusion. So there are two essential steps in a forward inference: searching a solution path and matching between the facts and rules. Final conclusions are made when no other rules are being fired.

Here we define the graph of all the paths for an inference system as a complete rule tree[14]. Searching solution paths from facts to final conclusions is searching paths from roots to leaves in the complete rule tree. We use a node neither a root nor a leaf to denote an intermediate conclusion, of which the amount is important to keep the molecular inference process going on. It will be discussed in detail in Section 3.5.

2 Implementation of forward inference

In this model, we use PpiI to run the forward inference. PpiI's has a special recognition site (RS)-a 5mer random sequence flanked by GAAC and CTC. If the recognition site is integral, PpiI will cut the molecule at four cleavage sites, 7/12 on the left side, and 13/8 on the right side, as shown in Fig. 1(a), and for more details, see [17]. We make the 5mer random sequence in RS encode a fact. In the following part, this portion is used as a sticky end to perform matching between a fact and its corresponding premise in a rule.

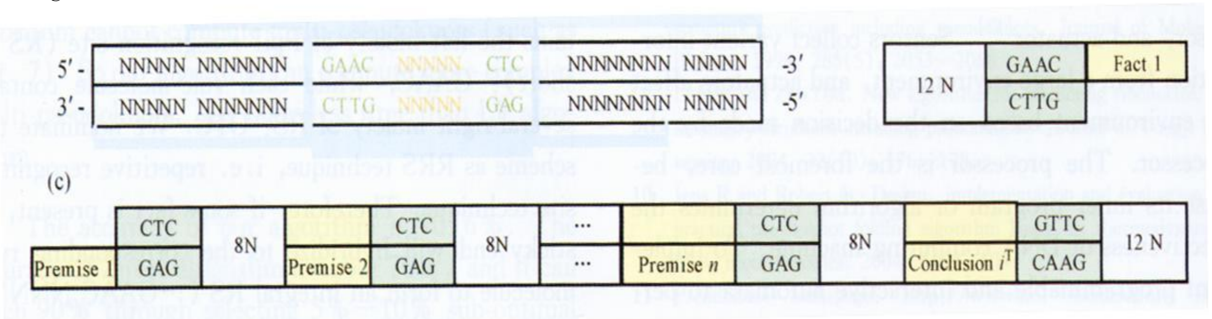

Fig. 1. Encoding of the facts and rules in knowledge base. (a) The cutting pattern of PpiI is 5'-7/12 GAAC NNNNN CTC 13/8-3'; (b) fact-molecule. A fact-molecule is a short dsDNA molecule with 12 bp random sequence, the left moiety of PpiI's recognition site GAAC/CTTG as well as a sticky end at 3' encoding a fact in the knowledge base; (c) dsDNA representing the rules in the knowledge base. The conclusion moiety on the right side of the rule molecule is comprised of the reverse sequence of the left RS as well as the reverse sequence of the conclusion.

2.1 Encoding of facts

A fact is encoded at 3'-sticky end in a dsDNA, which is called fact-molecule (Fig. 1(b)). The double-strand part of a fact-molecule contains a random sequence with 12 bp fixed length as well as the left moiety of the recognition site of PpiI, i.e. GAAC. The 5mer sticky end encodes the fact, which is complementary to its corresponding premise in the rule-molecule. To be concise, we define fact, and its corresponding premise has the same label.

2.2 Encoding of rules

A rule

IF (premise-1, premise-2, …, premise-n)
THEN (conclusion-i)

means that if the corresponding facts (fact-1, fact-2, …, fact-n) are present, then the rule is fired to make conclusion-i. We design a double-stranded rule-molecule to encode a single rule (Fig. 1(c)). Each rule-molecule has two parts—the premise moiety and the conclusion moiety. Premises and facts with the same label are encoded in 5mer-long complementary sequence. The structure of premise moiety is several repetition of {premise, right RS, 8N}, in which 8N means spacer with 8mer-long random sequence. The conclusion moiety is comprised of the reverse sequence (denoted by the upper label "T") of the left RS and the reverse sequence of the conclusion.

2.3 The forward inference process

As mentioned above, if all the premises in a rule are present, then the rule is fired to release its conclusion into water solution, and this molecule may in return match other rules as a newly produced input fact. In this way, the forward inference runs till no other rules are fired.

Put the rule-molecules and fact-molecules together. If one fact corresponding to the first premise of a certain rule is present, then this fact-molecule will hybridize to its counterpart in the rule-molecule and thus initiate the forward inference. After hybridization with ligase, they form an integral RS, thus PpiI cuts the rule-molecule to expose the next premise in form of a $3'$-sticky end. Provided that all the necessary facts are present, restriction enzyme will cut the rule-molecule all the way until it reaches the final part of the conclusion moiety. After releasing the conclusion, the fired rule is broken and no longer participates in the inference.

To symbolize the inference process via DNA computing, we introduce the following operators.

2.3.1 Defintion

[ ] and ( ) denote a double stranded and single stranded DNA molecule, respectively; $>$ and $<$ denote $3'$-sticky end on the upper strand and lower strand, respectively; Use $\mid$ to separate two neighboring functional sequence; ~ denotes the complementary sequence; $(\ )^{\mathrm{T}}$ denotes the reverse sequence; $\oplus^k$ denotes hybridization with ligase; $\perp_{PpiI}$ means use PpiI to cut the molecule; $\rightarrow$ means deducting the functional molecule for the next reaction.

2.3.2 Matching algorithm

Suppose the matching process between rule-i and its corresponding facts is at the jth time, $j=1$, $\cdots$, n. The fact-j molecule $[12N \mid lRS \mid f_j\rangle$ is going to match rule-i on the sticky end encoding premise-j

$$\langle p_j \mid rRS \mid 8N \mid \cdots \mid p_n \mid rRS \mid 8N \mid (c_i \mid lRS \mid 12N)^{\mathrm{T}}]$$

where we use f_j to denote fact-j, p_j to denote premise-j, and c_i to denote conclusion-i. lRS and rRS denote the left and right moiety of the recognition site, respectively, i.e. GAAC and CTC. Because the fact and premise that have the same label have complementary sequence, that is $f_j=\tilde{p}_j$, we have fact-j molecule matching rule-i molecule as follows, where the part $lRS \mid p_j \mid rRS$ represents an integral recognition site of PpiI.

$$\begin{aligned}
&[12N \mid lRS \mid f_j\rangle \oplus^k \langle p_j \mid rRS \mid 8N \mid \cdots \\
&\quad \mid p_n \mid rRS \mid 8N \mid (c_i \mid lRS \mid 12N)^{\mathrm{T}}] \\
&= [12N \mid lRS \mid p_j \mid rRS \mid 8N \mid p_{j+1} \mid rRS \mid 8N \mid \\
&\quad \cdots \mid p_n \mid rRS \mid 8N \mid (c_i \mid lRS \mid 12N)^{\mathrm{T}}] \\
&\perp_{PpiI} (5N) + \langle 5\tilde{N} \mid 7N \mid lRS \mid p_j \mid rRS \mid 8N \mid \\
&\quad p_{j+1}\rangle + \langle p_{j+1} \mid rRS \mid 8N \mid \cdots \mid p_n \mid rRS \mid 8N \mid \\
&\quad (c_i \mid lRS \mid 12N)^{\mathrm{T}}] \\
&\rightarrow \langle p_{j+1} \mid rRS \mid 8N \mid \cdots \mid p_n \mid rRS \mid 8N \mid (c_i \\
&\quad \mid lRS \mid 12N)^{\mathrm{T}}]
\end{aligned}$$

Similarly, when j goes from 1 to $n-1$, we have

$$[12N \mid lRS \mid f_n\rangle \oplus^k \langle p_n \mid rRS \mid 8N \mid (c_i \mid lRS \mid 12N)^{\mathrm{T}}] \rightarrow [12N \mid lRS \mid c_i\rangle$$

In this matching, we see that the output conclusion of rule-i $[12N \mid lRS \mid c_i\rangle$ has the same structure with the input fact. Therefore, it can be used in later inference process. Next we use a very simple example to describe the forward inference process in a more vivid way. Set the knowledge base composed of two rules written in the following symbolic form, and suppose that two facts P and R are present.

Rule 1: if P then Q

Rule 2: if Q and R then S

Then the inference process is

$$\begin{aligned}
&[12N \mid lRS \mid f_P\rangle \oplus^k \langle p_P \mid rRS \mid 8N \mid (c_Q \mid lRS \mid 12N)^{\mathrm{T}}] \rightarrow [12N \mid lRS \mid c_Q\rangle \\
&[12N \mid lRS \mid c_Q\rangle \oplus^k \langle p_Q \mid rRS \mid 8N \mid p_R \mid rRS \mid 8N \mid (c_S \mid lRS \mid 12N)^{\mathrm{T}}] \\
&\qquad \rightarrow \langle p_R \mid rRS \mid 8N \mid (c_S \mid lRS \mid 12N)^{\mathrm{T}}] \\
&[12N \mid lRS \mid f_R\rangle \oplus^k \langle p_R \mid rRS \mid 8N \mid (c_S \mid lRS \mid 12N)^{\mathrm{T}}] \rightarrow [12N \mid lRS \mid c_S\rangle
\end{aligned}$$

Later in Fig. 2 we explicitly describe this inference process. To check what final conclusions have been made, we use surface-based fluorescence detection. According to the complete rule tree, we array all the leaves on the surface in form of dsDNA with sticky end complementary to that of the corresponding conclusion molecules. Also we should tag fluorescence on the conclusion moiety of all the rules that are able to deduct leaves, i.e. the final conclusions. So when the inference process is over, put the surface into the water solution, thus the final conclusions of the inference will hybridize to their counterparts on the surface. Then check the fluorescence on the surface and read out what final conclusions have been made.

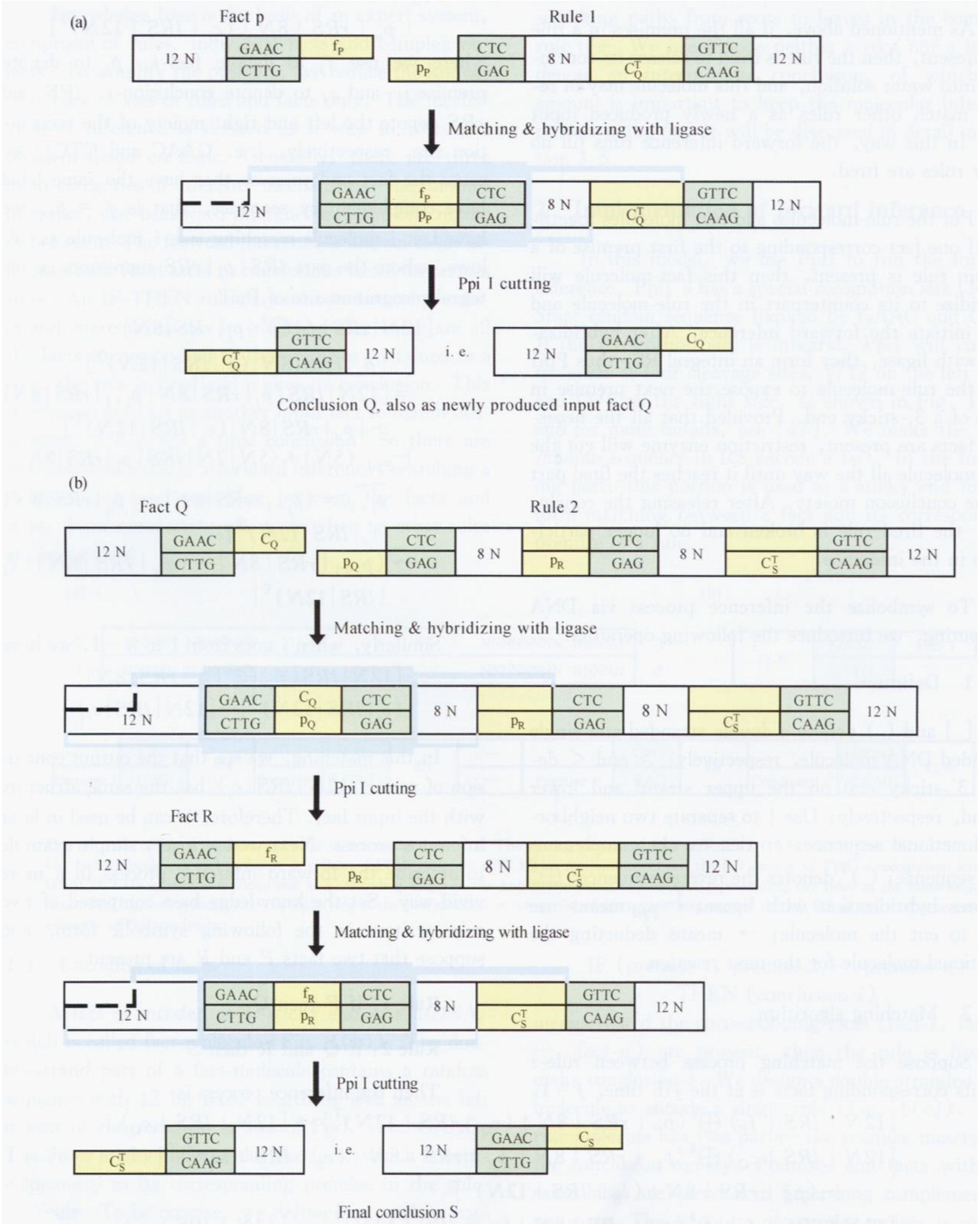

Fig. 2. The inference process. The forward inference is a cycle of matching, hybridizing and cutting by enzyme PpiI. The process is autonomous. When a fact-molecule matches its corresponding premise in a rule, then it hybridizes to the rule-molecule on the 3′ sticky end and thus an integral PpiI recognition site is formed. Enzyme PpiI cuts the rule-molecule to expose the next premise. The digesting process will stop if only the conclusion part is met. Then the conclusion of a fired rule will go into water solution as a newly produced input fact and participate in the next cycle of inference. (a) Matching between fact-P and rule 1 produces new input fact-Q; (b) two matchings between fact-Q and rule 2 form RS of PpiI and thus rule 2 is cut to expose the next premise-R. After fact-R and rule 2 match to each other, the final conclusion S is made.

2.4 Concentration analysis for the example in 2.3

There are two main factors that affect the amount of conclusions. One is the depth of inference; the other is the embranchment of the reactants in the process of inference. Firstly, we review the example in 2.3 to demonstrate the effect caused by single path inference. A concentration model is built to calculate the concentration of the resultants. To be concise, we do not take into account the time period for the reaction to reach equilibrium, and thus reduce this problem into simply considering the equilibrium concentration of reactants and resultants under ideal chemical reaction conditions.

In this example, each inference is implemented by hybridization and enzyme cutting. Take Rule 1 as an example of the basic inference process, it involves two reactions as follows

$$\text{hybridizing: } P + P\text{–}Q \rightleftharpoons PQ$$
$$\text{cutting: } PQ \rightleftharpoons P' + PQ' + Q$$

where $P\text{–}Q$ represents Rule1-molecule, PQ represents the product of hybridization with ligase, and Q is the conclusion of this inference. Since reactant P is the input fact, we suppose P is sufficient, and $P\text{–}Q$ is relatively insufficient. Define equilibrium coefficient $K_1=[PQ]/[P\text{–}Q]$, where $[PQ]$ is the concentration of resultant PQ, $[P\text{–}Q]$ is the concentration of $P\text{–}Q$ at the time when the reaction reaches its equilibrium. Define K_2 in the same way as K_1, $K_2=[Q]/[PQ]$, where $[Q]$ is the concentration of Q and $[PQ]$ is the same one in definition in K_1. So we have the concentration of Q satisfying

$$[Q] = [P\text{–}Q]_0 \frac{K_1K_2}{1+K_1+K_1K_2}$$

where $[P\text{–}Q]_0$ is the initial concentration of $P\text{–}Q$. We call it the reaction equation. As a result, after one step of inference, including one hybridization and one enzyme cutting, the amount of conclusion molecules decreases by $\frac{K_1K_2}{1+K_1+K_1K_2}$.

Further, we consider n steps of inference in a single path without embranchment. We suppose the equilibrium constants of all the hybridizations equal to K_1, and the counterparts of all the cutting reactions equal to K_2. Here K_1 and K_2 are mainly determined by the reaction temperature. Since the active temperature of the enzyme PpiI is about 310 K, the tube experiment should be performed at about 310 K. The pH and salinity of the solution are also important factors to the values of K_1 and K_2.

Therefore, in general, if a rule has n premises, it needs n steps of hybridization and cutting to produce its conclusion, of which concentration satisfies

$$[\text{conclusion}] = [\text{rule}]_0 \frac{K_1^nK_2^n}{1+(K_1+K_1K_2)(1+K_1K_2+\cdots+K_1^{n-1}K_2^{n-1})}$$

Secondly, we take the embranchment into consideration. Embranchment is the process that several different rules use one intermediate conclusion as an input fact according to the complete rule tree, and thus resulting in competitions for the intermediate conclusion molecules. If the competition occurs and makes the amount of conclusion molecules fall rapidly, we can use the amplification method in Section 3.5 to avoid the amount of conclusion from decreasing to an undetected level.

2.5 Discussion on the experimental issues of the inference model

If an inference system has only one path, we can easily design the initial amount of the fact-molecules and the rule-molecules. However, in a complete rule tree, when all possible inferences run on parallel in different paths, it will become complex to determine the exact initial amount of rule-molecules and fact-molecules. In this matter, two factors may help to determine the relative amount of these two kinds of reactants. One is the times of a reactant used in the inference process. Reactants that will be used more times will have a relatively higher initial amount than that of the others. Next is the inference depth of a reactant used in the complete rule tree, that is, reactants that will be used in deep inference depths will have a relatively smaller initial amount than that of the others.

Because we encode a fact and its corresponding premise in a 5mer-long complementary sequence, it will limit the scale of inference. The number of facts and conclusions together will be less than 512. It is determined by the cutting pattern of PpiI. Under this scale, when there are not many branches in the complete rule tree, we can simply mix the initial fact-molecules and rule-molecules together to run the forward inference. Otherwise, when the branches of inference are too many, each step of inference may result in a low yield of its conclusions. So in this case, we make the artificial control as the following steps (amplifying method):

Step 1: Before the inference, divide the complete rule tree into several levels $L_1, L_2, \cdots, L_n$.

At each level, we define its leaves as L-conclusions. Prepare n empty test tubes $T_1, T_2, \cdots, T_n$.

Step 2: For $k=1, \cdots, n$, select all the rules that will be used in L_k and put the corresponding rule-molecules into test tube T_k.

Step 3: Put the initial fact-molecules into T_1 and start to run the incomplete forward inference.

Step 4: For $k=1, \cdots, n-1$ carry on the following iterations.

(1) Pick up the L-conclusions of L_k in T_k by beads, and amplify them by PCR. Then put them back to T_k and thus form T_k'.

(2) Put T_k' into T_{k+1} and the inference is driven forward.

Step 5: The final conclusion of the forward inference is in T_n'. Use the method described in Section 2.3 and check what conclusions have been made.

By these several steps of amplification of some of the intermediate conclusions, the forward inference will continue and the yield of the final conclusions will be high enough to be detected.

3 Extended application of RRS technique to a triggering communication between automata

Studies[1—12] in these few years lead researchers to implement diverse DNA computing machines that are able to function in many problem-solving solutions. To us, it is of much interest to broaden the ability of the molecular computing machines by linking them together. However, the communication between different computing machines is not easy to solve. Recent work by Beyer and Simmel[18] demonstrated a molecular translator, which is able to translate an arbitrary input ssDNA into a functional output. It is interesting to find that if the output of the molecular translator has the same structure of the input molecular, then a 1∶1 triggering communication can be carried out. Here, we extend to implement a $m:n$ triggering communication between automata based on our model.

A 1∶1 triggering communication is that the output of one automaton can activate another automaton to run according to a certain communication path. Similarly, a $m:n$ triggering communication is that the outputs of m automata can trigger the other n automata to run according to certain paths or rules. Actually between two automata, this is a concatenation operation. But in complex network, the triggering communication can be $n:1$ or $1:n$. In this case, a forward inference based on rules (communication paths) seems necessary. To be concise, we exemplify a 2∶1 triggering communication between three automata. Theoretically, our model can be used to implement $m:n$ triggering communication, for any positive integer m and n.

3.1 Example

Suppose Benenson's automaton is used[1]. Define three automata A, B, and C, the outputs of A and B (shown as a, b) can trigger C to start running, the triggering signal is c. So the triggering path is: if a, b then c, which is defined by a rule-molecule $(a, b) \rightarrow c$. All of a, b, c have the same structure to the fact-molecule (Fig. 3(a)). Modify the automaton's accepting verification part to have the same structure to the conclusion moiety in our model; and add a triggering signal receiver at the very beginning of automaton (Fig. 3(b)). The triggering communication path is represented by rule-molecule $(a, b) \rightarrow c$, (Fig. 3(c)). So it is easy to see when automaton A makes its output a, and B makes output b, then they release a and b in form of newly produced input facts, which together with the rule-molecule $(a, b) \rightarrow c$ run the forward inference till an activating signal c is made. Finally, c activates automaton C by hybridizing on the c sticky end to form integral recognition site of PpiI. PpiI then cuts the automaton to expose its initial state (Fig. 3(d)).

4 Conclusion

In this paper, an algorithm to perform molecular forward inference via DNA computing has been described. By using a Class IIB restriction enzyme PpiI as well as introducing a novel repetitive recognition site (RRS) technique, an autonomous forward inference mechanism has been built.

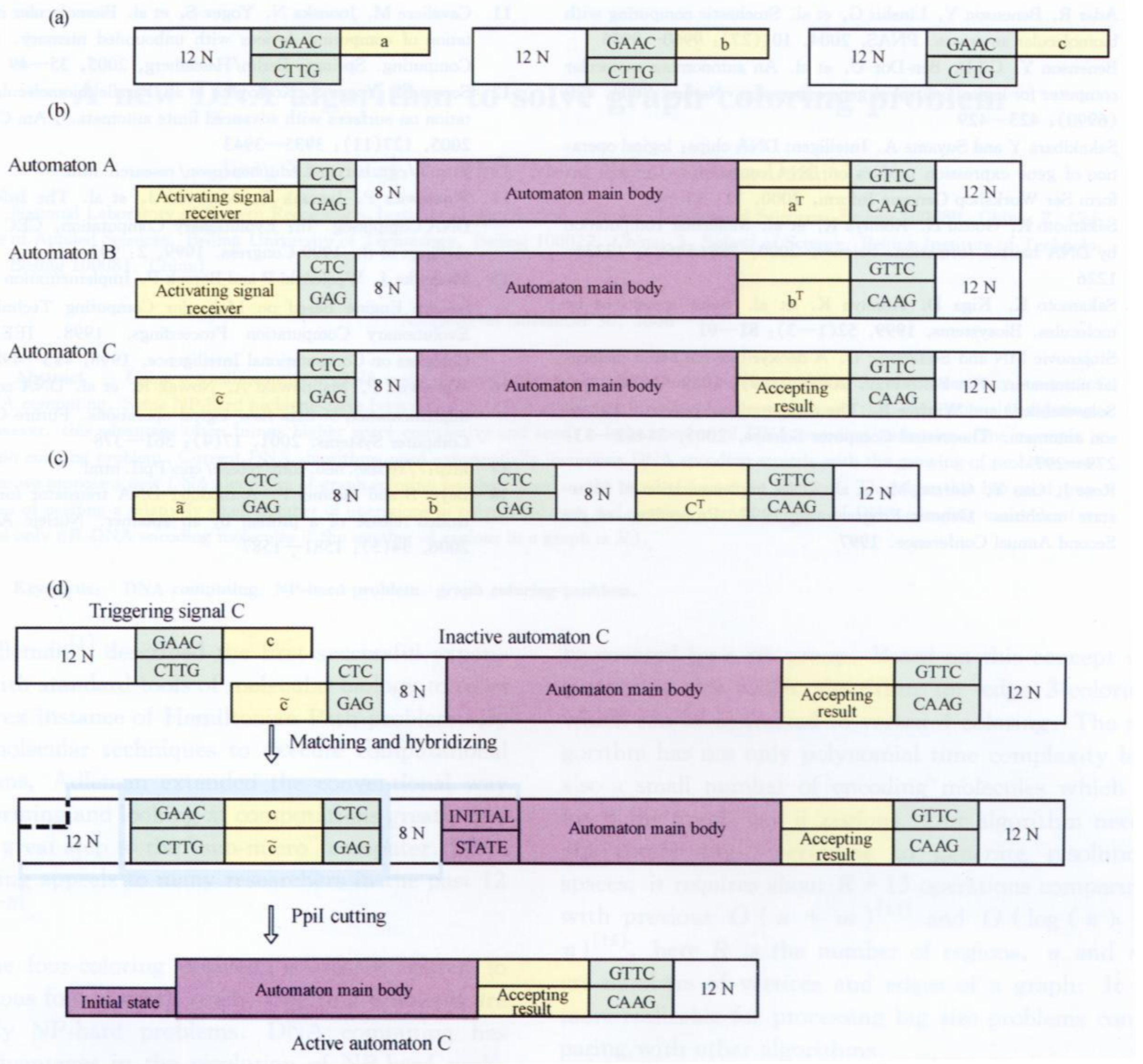

Fig. 3. Extension application of our model to a triggering communication between automata along certain communication path. (a) The conclusions made by automata A, B and C; (b) the modifications to the Benenson's automaton in order to add in a triggering mechanism; (c) rule-molecule represents a triggering communication path $(a, b) \rightarrow c$; (d) the triggering communication process is the same to the forward inference mentioned above. When rule $(a, b) \rightarrow c$ is fired, a triggering signal c is released. Then c hybridizes to the inactive automaton C on the sticky end, which encodes a triggering signal receiver. Finally, there forms an integral recognition site and thus PpiI cuts automaton C to expose its initial state. Automaton C then becomes active to run.

When one rule-molecule in knowledge base captures the facts according to each of its premises, the rule is fired and produces its conclusion, which may match the premise on other rules as a new input fact, that is, one output conclusion of a certain rule may be the input fact of other rules. In early studies, one rule can only match one kind of conclusion of other fired rules. The advantage of this model is that a rule is able to match more than one conclusion of other fired rules.

In the end, we further extend our model to implement a triggering communication between different automata, i.e. using the output of one automaton to activate another automaton. This autonomous molecular forward inference mechanism ensures $m:n$ triggering communication network, providing a kind of linkage between several molecular computing machines to perform complex task.

References

1 Benenson Y, Paz-Elizur T, Adar R, et al. Programmable and autonomous computing machine made of biomolecules. Nature, 2001, 414: 430—434

2 Benenson Y, Adar R, Paz-Elizur T, et al. DNA molecule provides a computing machine with both data and fuel. PNAS, 2003, 100 (5): 2191—2196

3 Adar R, Benenson Y, Linshiz G, et al. Stochastic computing with biomolecular automata. PNAS, 2004, 101(27): 9960—9965

4 Benenson Y, Gil B, Ben-Dor U, et al. An autonomous molecular computer for logical control of gene expression. Nature, 2004, 429 (6990): 423—429

5 Sakakibara Y and Suyama A. Intelligent DNA chips: logical operation of gene expression profiles on DNA computers. Genome Inform Ser Workshop Genome Inform, 2000, 11: 33—42

6 Sakamoto K, Gouzu H, Komiya K, et al. Molecular computation by DNA hairpin formation. Science, 2000, 288(5469): 1223—1226

7 Sakamoto K, Kiga D, Komiya K, et al. State transitions by molecules. Biosystems, 1999, 52(1—3): 81—91

8 Stojanovic MN and Stefanovic D. A deoxyribozyme-based molecular automaton. Nat Biotechnol, 2003, 21(9): 1069—1074

9 Soloveichik D and Winfree E. The computational power of Benenson automata. Theoretical Computer Science, 2005, 344(2—3): 279—297

10 Rose J, Gao Y, Garzon M, et al. DNA implementation of finite-state machines. Genetic Programming 1997: Proceedings of the Second Annual Conference. 1997

11 Cavaliere M, Jonoska N, Yogev S, et al. Biomolecular implementation of computing devices with unbounded memory. In: DNA Computing. Springer Berlin/Heidelberg, 2005, 35—49

12 Soreni M, Yogev S, Kossoy E, et al. Parallel biomolecular computation on surfaces with advanced finite automata. J Am Chem Soc, 2005, 127(11): 3935—3943

13 http://cgr.harvard.edu/benenson/research.htm

14 Wąsiewicz P, Janczak T, Mulawka J, et al. The Inference via DNA Computing. In: Evolutionary Computation, CEC 99. Proceedings of the 1999 Congress. 1999, 2: 993

15 Mulawka J, Węgleński P and Borsuk P. Implementation of the Inference Engine Based on Molecular Computing Technique. In: Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, 1998, 493—498

16 Wąsiewicz P, Malinowski A, Nowak R, et al. DNA computing: implementation of data flow logical operations. Future Generation Computer Systems, 2001, 17(4): 361—378

17 http://rebase.neb.com/rebase/enz/PpiI.html

18 Beyer S and Simmel F. A modular DNA translator for the controlled release of a protein by an aptamer. Nucleic Acids Res, 2006, 34(5): 1581—1587