Ligation module for *in-vitro* selection in DNA computing

Danny van Noort[1,2], In-Hee Lee[1], Laura F. Landweber[2] and Byoung-Tak Zhang[1]
[1]Biointelligence Lab., School of Computer Science and Engineering, Seoul National University, San 56-1, Sinlim-dong, Gwanak-gu, Seoul 151-742, Korea
[2]Ecology and Evolutionary Biology, Princeton University, Princeton, NJ 08544 USA

ABSTRACT.

In this paper a classical AI problem is proposed to be solved by DNA computing: theorem proving. Since the complexity grows exponentially with the size of the problem, the solving process should be done in parallel. Massive parallelism is one of the advantages of DNA computers. It will be shown that the resolution refutation proof can be readily implemented by DNA hybridisation and ligation. Microreactors lend themselves to a relatively simple implementation of DNA computing. Not only is the design of the DNA critical for the success of the system but also the architecture of the microfluidic structure. Here the DNA performs the computation, while the microfluidics aids the biochemical steps necessary to manipulate the DNA, *i.e.* hybridisation and ligation.

Keywords: theorem proving, DNA computing, ligation, PDMS valves

1. INTRODUCTION

Biomolecular computing involves a multi-disciplinary approach consisting of molecular biology, microsystems, sensing and information technology. It uses biological macromolecules, such as DNA, to perform selections, simulating the digital information processing procedures. The largest problem solved to date is a 20-variable instance of a SAT problem [1]. Alternative approaches incorporated, among others, RNA [2], self-assembly [3] and pipetting robots [4].
Microflow structures provide control over the flow of solutions, *i.e.* the flow of information [5]. The advantages of this technology are, for example, the use of small volumes of biological solutions, which are in the nanoliter range, and increased reaction rates due to reduced diffusion time. Furthermore, microflow structures can be designed to be problem-specific or re-configurable [6, 7]. By using valves and pumps in the fluidic network, the computer architecture can be problem independent [8] and autonomous.
In the past, many researchers have utilised the massive parallelism of DNA computers to solve hard computational problems [9, 10, 11, 3, 7, 18]. Recently however, several research groups have proposed DNA computing methods for logic reasoning [12, 13, 14, 15, 6, 10, 16, 17]. Here we look at a more traditional AI problem to solve. Theorem proving is a method of logic reasoning from the propositional logic. It has a variety of applications, including diagnosis and decision making [16, 17, 8, 11].
A way to prove a theorem is by resolution refutation which is a general technique using a given set of axioms and rules. However, if the proposition becomes to complex or the number of axioms to large, the problem grows exponentially over time. To overcome this drawback, parallel theorem proving methods have been proposed [18, 19, 20, 5, 9, 15]. However, these theories do not overcome the difficulties inherent to silicon-based technology. DNA molecules, however, could solve this problem. In this paper, we give a short introduction to theorem proving be resolution refutation, a description of the experimental procedure and the design of the microsystem to be used.

2. THEOREM PROVING BY RESOLUTION REFUTATION

Theorem proving is a method for automated reasoning [16, 17]. In this section, propositional logic, resolution principle and resolution refutation will briefly be described.
Propositional logic formula consists of Boolean variables and logic connectives. A Boolean variable is a variable which can be either T (*TRUE*) or F (*FALSE*). Among the basic logic connectives are $\wedge$ (*and*, logic product), $\vee$ (*or*, logic sum), $\neg$ (*not*, negation) and $\rightarrow$ (*implication*). A Boolean variable or its negation is called a *positive literal* resp. *negative literal*. It has been proven [21] that any connective can be expressed by using only the $\wedge$, $\vee$, and $\neg$ operators. For example, $A \rightarrow B$ can be written as $\neg A \vee B$ for any Boolean variables A and B. To prove theorems using resolution refutation, every formula must be expressed in a so called clause form. A clause form in propositional logic is defined as follows:

(clauseform) := (clause) $\wedge$ (clause) $\wedge \cdots \wedge$ (clause)
(clause) := (literal) $\vee$ (literal) $\vee \cdots \vee$ (literal)

A clause with no literal is called an empty clause.
The basic idea of refutation is to show that the negation of the conclusion is inconsistent with the premises. If true, an empty clause will remain. The outcome of the resolution is called a *resolvent*.

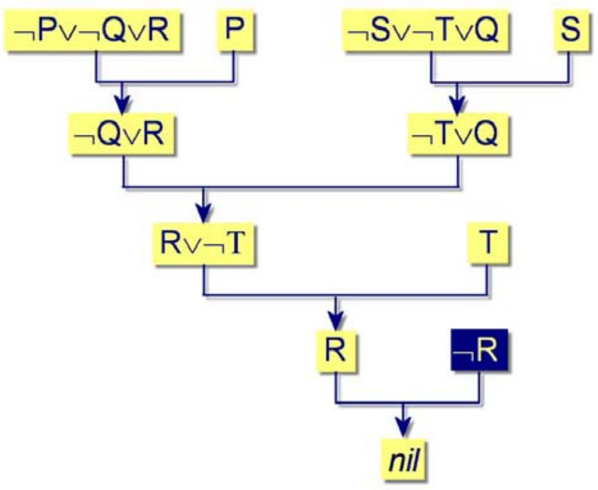

Figure 1. A proposed problem in theorem proving using resolution refutation. Here R is the premise.

Figure 1 shows an example of theorem proving by resolution refutation. The set of formulas $\Delta = \{P \wedge Q \rightarrow R, S \wedge T \rightarrow Q, S, T, P\}$ is given. It should be proven that the premise R is consistent with Δ. Δ is converted to the *conjunction normal form* by using associative and distributive laws, after which the set of clauses are written as $\{\neg P \vee \neg Q \vee R, \neg S \vee \neg T \vee Q, S, T, P\}$. The negation of R, *i.e.* $\neg R$, is added to prove the possible inconsistency. Each box in Fig. 1 contains one clause. The clauses and their resolvents are connected with arrows. Resolving on P from $\neg P \vee \neg Q \vee R$ and P results in $\neg Q \vee R$. This process can be continued until an empty clause is produced, denoted by the symbol *nil*.
The above theorem proving process becomes more complex as the number of formulas grows. In propositional logic, the literal to resolve on must be decided. Therefore, if there are n different literals, there are $n!$ different theorem proving processes with $n! = O(n^n)$ according to Stirling's approximation. Thus, the complexity of proof grows exponentially with the number of literals. As already has been mentioned, one way to overcome this problem of complexity is to use a representation in DNA molecules.

3. THEOREM PROVING USING DNA

In [22], a method using representation in DNA molecule is proposed to solve the theorem proving problem effectively. A brief description to the method is given here before introducing the design of the microreactor system for the method. (I inserted this paragraph to make sure that the scope and main result of this paper is the design of microreactor system not the molecular theorem proving method. I also think this section is too long compared to the next section which is about the microsystem – the main content.)
In this method [22], the literals, and therefore the formulas, are represented by short single stranded DNA sequences. The negations of the literals are represented by the complementary sequence of the positive literals.

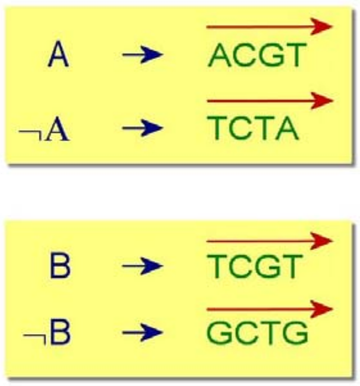

Figure 2. An example coding for a Boolean variable and its negation (The arrows are from 5' to 3').

Each literal has the same sequence length, in this case 4 nucleotides. An example of two Boolean variables and their negations are shown in Fig. 2. And each clause is represented by the concatenation of the sequences for the literals contained in the clause.
Resolution of a variable between two clauses is represented by the hybridisation of the two short DNA sequences corresponding to that literal in each clause. For example, when a variable *v* is resolved from clauses *A* and *B*, the DNA sequences and its complementary corresponding to *v* or $\neg v$ in each clause hybridise. Therefore, if the resolvent is an empty clause, none of the DNA sequences representing the literals will remain single-stranded. Afterwards the nicks in the hybridisation products are ligated. The verification of the existence of a blunt ended double stranded DNA molecule with a defined size, which represents the empty clause, proves refutation has worked. Finally, the result is examined by PCR and gel electrophoresis [22].
The experimental steps are summarized as follows (see Fig. 3):

1. Mix the clauses.
2. Hybridise the DNA sequences to perform resolution.
3. Ligate the hybridised products for easier screening of the proof.
4. Perform PCR amplification of the ligation products. In this step, only the ligation product which is a valid proof can be amplified, due to the primer choice.
5. Perform gel electrophoresis to detect the proof, if any.

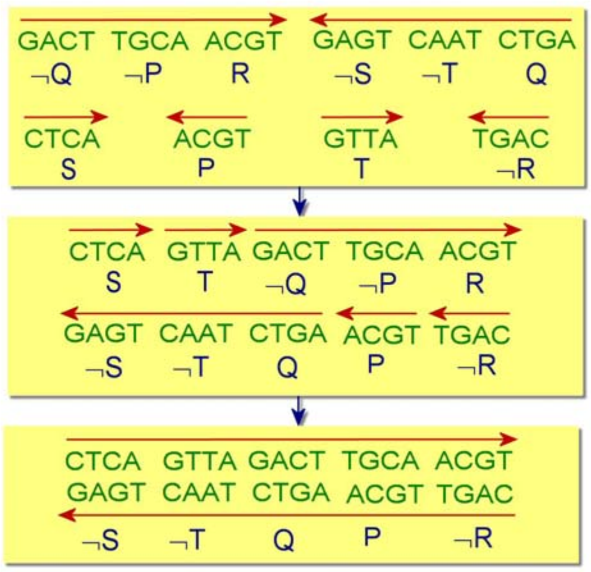

Figure 3. Theorem proving by refutation, where each literal is represented by a 4 nt sequence. Only when all the literals and their negations are present will there be a blunt ended double stranded DNA. (The arrows are from 5' to 3'.)

4. THE MICROSYSTEM

To perform step 1, 2 and 3 as described above in microfluidics a mixing device is needed in which DNA and ligation enzymes can be mixed. As is well known in microfluidics, mixing is a well-known problem as laminar flows only mix by diffusion. However, with the PDMS (Polydimethylsiloxane) valve technology developed in Steve Quake's lab [23], it is quite simple and very effective to construct a mixing chamber. These valves can be used to regulate the inputs and output (see Fig. 4a). Furthermore, by also using the PDMS valves, a peristaltic pump was fabricated to produce which pumps the solution around a circular channel (Figs. 4c and d). Due to the parabolic flow profile in a pressure driven flow, mixing is imminent.
Two input channels are needed for a separate input of DNA and the ligation enzymes. These channels can be closed by a valve just before the mixing chamber. By opening these valves, the chamber can be filled. By closing the valves of the input and output channels, the mixing chamber is sealed off from the rest of the system and mixing can commence. The pump section is made of 3 PDMS valves which circulate the solution within the mixing chamber. At high speed, only a few seconds is necessary for mixing. After mixing, the output channel is opened and the solution can flow over functionalised beads. Figure 5b shows the complete structure of the ligation module in PDMS. After the ligation there will still be products left with so called sticky ends, *i.e.* double stranded DNA molecules with a single stranded overhang, caused by missing literals and therefore not representing the correct answer to the problem. These can be filtered out by using functionalised beads that recognise the patterns of 4 nt long DNA sequences. To do so, the output of the mixer is opened and the solution flows into a column packed with the beads, held in place by another set of valves [23] (Fig. 4a). The degree to which the channel is closed depends on the pressure applied to the valves. By allowing the valve to close partially, beads are captured while the flow could continue, turning these valves into bead barriers.
The device was made in 2 layers of PDMS (Sylgard 184, Dow-Corning, MI), the fluidic and the pneumatic layer. A thin layer of PDMS, for the valves [24], with a mixing ratio of 1:20 (curing agent to base) was spin-coated on a 2" silicon wafer containing the master with the fluidic channels to obtain a thickness of approximately 40 µm. A thicker layer for the pneumatic layer, about 10 mm, of PDMS with a ratio of 1:5 was poured onto the mask in a petri dish.

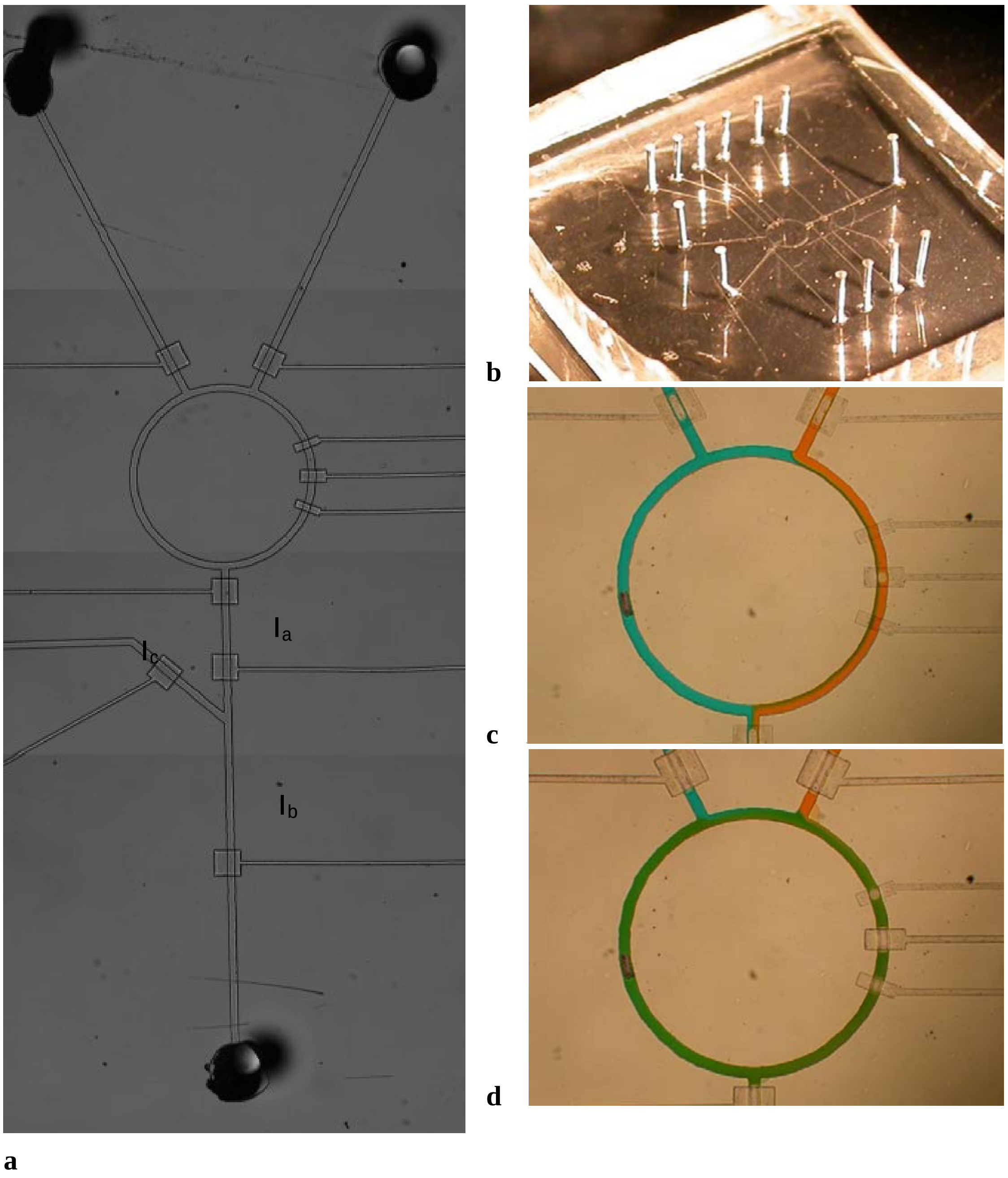

Figure 4. *(a) An overview of the ligation module, containing 3 inputs, 9 PDMS valves and 1 output. Before the output is the filtering section, where beads are retained between 2 valves (I_a and I_b), while the 3rd input (I_c) is used as bead delivery channel. (b) shows the module in PDMS. (c) The mixer with 2 dyes in unmixed stage. (d) after the peristaltic pump (the group of 3 valves) has operated, the dyes are mixed.*

After degassing and curing for 35 minutes at 65 °C, the thick layer was cut out and small holes were punched (Technical Innovations Inc., TX, USA) for the inlets to the PDMS valves. After cleaning the layers with action and iso-propanol, the channel layer was aligned with the valve layer under a stereo-microscope and replaced in the oven for at least 2 more hours. Hereafter the fluidic inlets and outlets were punched. Tubing (Fisher Scientific, PA, USA) was connected via small metal tubing (New England Small Tube Corporation, NH, USA) inserted in the inlets to pressurized syringes (5 PSI) and the PDMS valves were operated with pressurized air (30 PSI) and solenoid valves (Lee Co., USA) controlled with LabVIEW (National Instruments, USA) with a DAC-board (NI PCI-6533, National Instruments). Transistor switching boards (Comfile Technology, South Korea) where used to switch the 12V solenoids with a 5V TTL-signal. The pressured air was obtained from the laboratory's outlet and down regulated using mini pressure regulators. When pressure is applied, the membrane is pushed into the underlying channel, effectively blocking the flow.
The results of the hybridisation and ligation in the microfluidic system will be published elsewhere.

5. CONCLUSION

Microfluidic systems have a great advantage in DNA computers. The biochemical reaction rates are much faster in test tubes, while the amount of consumables is in the nanoliters. Furthermore, methods were presented for the encoding of clauses with DNA molecules with which theorem proving problems could be solved, by using simple biochemical techniques. And with the presented method the number of experimental steps does not vary with the problem size. The presented ligation module will prove to be a valuable asset to miniaturising the chemical lab and to DNA computing problems incorporating ligations.

6. ACKNOWLEDGEMENT

The author wishes to acknowledge the support from DARPA award F30602-01-2-0560 to Laura F. Landweber and NSF award 0121405 to Lydia L. Sohn and Laura F. Landweber. Furthermore the support from the Molecular Evolutionary Computing (MEC) project of the Korean Ministry of Commerce, Industry and Energy, and the National Research Laboratory (NRL) Program from the Korean Ministry of Science and Technology is acknowledged. EEB at Princeton University and the ICT at Seoul National University provided the research facilities.

REFERENCES

1. Braich, R. S., Chelyapov, N., Johnson, C., Rothemund, P. W. K., and Adleman, L. (2002) *Solution of a 20-Variable 3-SAT Problem on a DNA Computer.* Science **296**, 499-502.
2. Faulhammer, D., Cukras, A. R., Lipton, R. J. & Landweber, L. F. (2000) *Molecular computation: RNA solutions to chess problems.* PNAS **97**, 1385-1389.
3. Winfree, E. (2000) *Algorithmic Self-Assembly of DNA: Theoretical Motivations and 2D Assembly Experiments.* Journal of Biomolecular Structure & Dynamics **11**, 263-270.
4. Suyama, A. (2002) *Programmable DNA computer with application to mathematical and biological problems.* Preliminary Proceedings, Eigth International Meeting on DNA Based Computers, June 10-13, 2002, Japan, 91.
5. van Noort, D., Wagler, P. and McCaskill, J. S. (2002) *The role of microreactors in molecular computing.* Smart Mater. Struct. **11**, 756-760.
6. van Noort, D., Gast, F.-U. and McCaskill, J. S. (2001) *DNA computing in microreactors.* LNCS 2340, 33-45.

7. McCaskill, J. S. (2001) *Optically programming DNA computing in microflow reactors.* Biosystems **59**, 125-138.
8. van Noort, D. (2004) *A programmable molecular computer in microreactors*. In press, LNCS.
9. Hagiya, M., Arita, M., Kiga, D., Sakamoto, K., and Yokoyama, S. (1997) *Towards parallel evaluation and learning of Boolean μ-formulas with molecules.* Preliminary Proceedings of the Third DIMACS Workshop on DNA Based Computers, 105–114.
10. Lipton, R.J. (1995) *DNA solution of hard computational problem*. Science **268**:542–545.
11. Winfree, E. (1998) *Algorithmic self-assembly of DNA*, Ph.D. Thesis, California Institute of Technology.
12. Kobayashi, S. (1999) *Horn clause computation with DNA molecules*. Journal of Combinatorial Optimization **3**:277-299.
13. Mihalache, V. (1997) *Prolog approach to DNA computing.* Proceedings of the IEEE International Conference on Evolutionary Computation, IEEE Press, 249–254.
14. Uejima, H., Hagiya, M., and Kobayashi, S. (2001) *Horn clause computation by selfassembly of DNA molecules*. Preliminary Proceedings of the Seventh International Meeting on DNA Based Computers, 63–71.
15. Wasiewicz, P., Janczak, T., Mulawka, J.J., and Plucienniczak, A. (2000) *The inference based on molecular computing.* International Journal of Cybernetics and Systems **31**(3):283–315.
16. Luger, G.F. and Stubblefield, W.A. (1993) *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. 2nd Ed., Benjamin/Cummings.
17. Nilsson, N.J. (1998) *Aritificial Intelligence: A New Systhesis*, Morgan Kaufman Publishers Inc.
18. Hasegawa, R. (1994) *Parallel theorem-proving system: MGTP.* Proceedings of Fifth Generation Computer System.
19. Lusk, E.L. and McCune, W.W. (1998*) High-performance parallel theorem proving for shared-memory multiprocessors.* http://wwwfp.mcs.anl.gov/ ~lusk/papers/roo/paper.html.
20. Suttner, C. (1997) *SPTHEO - A parallel theorem prover.* Journal of Automated Reasoning **18**(2):253-258.
21. Fitting, M. (1942) *First-Order Logic and Automated Theorem Provin.* Springer-Verlag New York Inc.
22. Lee, I-H, Park, J-Y, Jang, H-M, Chai, Y-G, and Zhang, B-T (2003) *DNA Implementation of Theorem Proving with Resolution Refutation in Propositional Logic*, LNCS, vol. 2568, 156-167.
23. van Noort, D. and Zhang, B-T (2004) *PDMS valves in DNA computers*. SPIE International Symposium on Smart Materials, Nano-, and Micro-Smart Systems, 12-15 December 2004, Sydney, Australia.
24. Marc A. Unger, Hou-Pu Chou, Todd Thorsen, Axel Scherer, Stephen R. Quake (2000) *Monolithic Microfabricated Valves and Pumps by Multilayer Soft Lithography*, Science **288**, 113-116.