

On Self-Assembling Graphs *in vitro*

The Molecular Computing Group

Max H. Garzon, Rusell J. Deaton, Ken Barnes, J.A. Rose
mgarzon@memphis.edu, rjdeaton@memphis.edu

The University of Memphis
Memphis, TN 38152-6429 U.S.A.

Abstract

In this paper we identify a new type of structures that can be assembled *in vitro* by self-regulating molecular processes. This type of structure, the so-called automatic graphs, is a class of highly symmetric graphs Cayley graphs that have the same local appearance from every node and yet admit a finite-state navigator set that can discriminate global properties of the graph. We show how the navigator can be implemented as a molecule that directs the self-assembly process to build such graphs as nanostructures in DNA molecules. Cayley graphs form a class of highly regular structures that can, in principle, be used as read-only memories to support universal computation, even of super Turing type.

1 Introduction

Nearly five years after Adleman's paper [1] established molecular computation as a feasible alternative to solve computational problems, a new direction seems to be emerging to better cope with the difficulties posed by molecular computation and better exploit the nature of biomolecules. Inspired by Seeman's experiments in nanomolecular design (see e.g., [24]) and Winfree's attempts to implement universal computation devices such as cellular automata and Turing machines *in vitro* [27, 28, 29, 30], renewed efforts have been directed at building complex molecular structures as a means

to further understand the computational capabilities of biomolecules. Other efforts in that direction include Jonoska-Karl [15] study of 3D graph structures for 3-SAT and 3-VERTEX COLORABILITY.

In this paper we identify a new type of structures to which feasible state machine protocols can be applied for self-assembly *in vitro* by self-regulating molecular processes. The regular structures are the so-called automatic graphs. This structure guarantees that the protocol will actually form fairly complex graphs with a good degree of reliability. Cayley graphs are highly symmetric graphs that have the same local appearance from every node. Yet, they admit a navigator (a finite-state machine) ensuring that the graphs can be assembled from a simple seed by a relatively simple repetitive process while ensuring that the graph will be completed with a high degree of reliability. Furthermore, these structures can, in principle, be extended and used as read-only memories to support universal computation.

This remainder of the paper is organized as follows. In section 2 we give brief definitions of molecular protocols. Next, we describe briefly finite-state machines, Cayley graphs, and automatic graphs in section 3. In section 4 we show a general procedure to implement fsm's capable of navigating automatic graphs for DNA self-assembly. In section 5 we describe one implementation protocol based on retro-viruses. Finally, in section 6 we discuss some implications

for general-purpose computation.

2 Molecular Computation

Molecular Computing problems are solved in three steps: *encoding* that maps the problem onto DNA strands, *molecular operations* that perform the basic core processing, and *extraction/detection* that makes the results visible to the naked eye. A typical set up consists of a multiset of molecules in a tube that encode the initial set up of the computation. The key components of each molecule are nucleic acid bases **A**, **T**, **G**, **C**, which bind according to the Watson-Crick complement condition. Once the encoding of the inputs of a problem have been chosen, the results of a DNA experiment are essentially determined up to laboratory execution. If the encodings are prone to errors, the experiment can be repeated any number of times and always provide the same (erroneous) result. A critical issues to address in molecular computing must therefore be not only the set up and protocol to perform a computation, but also built-in measures and an analysis to make sure that the possible mishaps and errors that may thwart the experiment will not be present.

3 Cayley graphs and Automatic Groups

We quickly review finite-state machines and their relation to the type of graphs dealt with in this paper. A fsm M is a 5-tuple $M := \langle Q, \Sigma, \delta, q_0, F \rangle$ consisting of an alphabet Q of ‘memory configurations’ with special state q_0 , an alphabet Σ of input symbols, a transition function $\delta : Q \times \Sigma \rightarrow Q$, and, finally, a distinguished subset $F \subseteq Q$ of *final* states. An fsm M begins in its initial state q_0 at the leftmost symbol a_1 of an input string. Thereafter, M computes by performing a series of atomic moves until it has exhausted the input string x past its rightmost symbol. An atomic move consists of sensing the

current input symbol a under the finite control, shifting the state of the finite control to state $\delta(p, a)$ and moving on to the next symbol in x (or the end of x). Finite-state machines can be used as pattern recognition devices that detect the presence of global features in an input just by looking at local features/symbols in it.

We propose to use fsm’s as devices that navigate in the sea of molecules in the test tube to determine, based on input patterns determined by the presence of certain molecules, a prescribed action to glue pieces of a partially built graph towards the completion of a self-assembly process. A *Cayley digraph* $\mathcal{C} = (V, \mathcal{A})$ admits a global positioning system defined by an algebraic structure (a group G) that coordinates its vertices, plus a homogeneous set of arcs (directed edges) corresponding to a set of generators X of the group G . The neighbors of a given vertex i are vertices of the form $i * A$ where A runs over the generators in X and $*$ is the group’s operation. A Cayley graph is homogeneous, i.e., an observer standing at a vertex cannot tell his position exactly just by looking at the neighborhood since any two neighborhoods look the same everywhere (isomorphic). For example, a cyclic group with n elements gives rise to a ring of nodes. A cartesian product of two of these groups yields the Cayley graph represented by a discrete lattice on the surface of a 2D torus. The Cayley graph of the cartesian product of n copies of \mathbf{B} , the additive integers modulo 2, with the standard generating set is a finite hypercube of dimension d on 2^d vertices, denoted \mathbf{B}^d . Cayley graphs are usually given by presentation consisting of *relators*, the closed loops in the graph so that every other closed loop can be obtained by gluing together copies of the basic relators shifted in some appropriate way. An example $Z_3 \oplus Z_2$, is shown in Figure 1. Plenty of other Cayley graph examples can be found in the classic book by Coxeter-Moser [6] and also Grossman-Magnus GM.

A special class of Cayley graphs and groups, so-called *automatic*, are characterized by the property that their Cayley graphs can be *nav-*

igated by means of finite-automata. We follow Baumslag’s review [4] to briefly make some details precise.

Definition 3.1 *A Cayley graph Γ , in d generators is automatic if there exists a set of $d + 2$ finite-state automata M , $M_{=}$, and M_{a_i} ($i = 1, \dots, d$), such that:*

1. *the input of M_{Γ} consists of all strings over the alphabet X^{\pm} of generators and their inverses, and its output is a set $L(M)$ of strings whose values comprise all of G ;*
2. *the input of the so-called equality checker $M_{=}$ is $\{(u, v)u, v \in L(M)\}$, and the output of $M_{=}$ consists of $\{(u, v)u, v \in L(M), u =_{\Gamma} v\}$, where the notation $=_{\Gamma}$ indicates equality of u and v as vertices in Γ ;*
3. *the input of the so-called multiplication checker M_a for a generator a is a string describing a pair (u, v) (u on top of v). M_a decides whether or not $1 u =_{\Gamma} v$ (as vertices on the graph).*

In other words, a Cayley graph is automatic if there is a machine M_{Γ} (i.e., a finite-state automaton) which provides a list of words representing its vertices, a machine which checks for equality, and a bunch of machines M_a ’s which check whether words represent adjacent vertices, i.e., differ by right multiplication by a generator a . Upon the graph, this means that two copies of a finite state machine M_a started at the same origin vertex can navigate from vertex to vertex following a given input course given by two words u and v , and decide whether the two terminal vertices are nearest neighbors. The existence of such machines turns out to be independent of the choice of the finite set X of generators of a given group G . A set of such machines is termed an *automatic structure* for the Cayley graph Γ . The direct product of two automatic groups is again automatic. A fairly complete treatment of automatic groups can be found in Epstein et al’s [9].

We present next a protocol to build automatic Cayley graphs in DNA molecules.

4 Building Cayley Graphs with Molecules

The automatic structure of a Cayley graph suggests a control mechanism to direct the general construction of the graph at the nanolevel in the tube. A molecular implementation of fsm’s has been given in several places, for example, Hagiya *et al.* [14, 28] and Garzon *et al.* [11]. Fairly efficient and reliable protocols to implement finite-state machines that have been tested in the lab and proved to be feasible in [14, 28]. First, whiplash PCR consists of thermal cycles that are easily automated while performing in parallel a large number of state transitions. Second, the transitions are self-controlled, i.e., once initiated, they stop themselves without explicit human intervention. The question is whether these protocols suffice to implement an automatic structure in the tube that would serve as a self-assembly procedure to build the graphs.

To date, self-assembling process in molecular protocols aimed at generating complex structures in DNA have relied essentially on the structure of the strands (3-way junctions and double-crossed over molecules in [27], for example) and on a good solution to the encoding problem. Good encodings guarantee that mismatched hybridizations will not occur, and may even guarantee that certain hybridizations will be recursively formed that effect the self-assembly process. It is unlikely, however, that given the difficulty of the encoding problem [7, 8], good encodings can actually be produced, or even exist, that alone guarantee that complex structures to be built with any high degree of reliability. In particular, in building a graph with biomolecules, portions of the graph can be formed, but finding a finished graph based on hybridization affinity alone is unlikely.

What is needed are built-in conditions in the protocol to ensure that the construction is com-

pleted as desired. The automatic structure of the Cayley graph can serve that purpose well. The homogeneity of the graph will guarantee two conditions. First, that the action of M_a^{edge} is repetitive and performed by a simple mechanism; second, that it remains good regardless of whether it is performed; and third, that the protocol guarantees that the overall architecture of the Cayley graph is really being obtained at the end of the protocol. Combined with appropriate encodings that are known to be very successful when actually implemented in the lab, an experiment is being performed to verify the self-assembling property of these graphs.

We can devise a molecule M_a that represents the fsm M_a in Def. 3.1, for each generator a , and which checks whether two words differ by a generator a , i.e., are nearest neighbors by an edge labeled a . M_a can be designed to have sticky ends complementary to symbols a and b so that their physical presence nearby will trigger a transition. If the fsm M_a sees the vertices as neighbors, then the polymerase reaction is enable and generates a new edge connecting the two neighboring vertices a and b . If the new state is not accepting, the polymerase reaction will simply use the newly created piece to add a new vertex and edge to the partially built graph and so complete one more step in the self-assembly process. In either case, the fsm molecule will leave the newly created edge behind and detach itself in a new state to float around looking for other sites to add one more step to the construction. This approach can be implemented in both DNA and RNA molecules. In the short space left, we describe an implementation that could possibly take place *in vivo* using retroviruses.

5 An implementation using Retroviruses

We describe now a method to implement this self-assembly. The inspiration for this method is a retrovirus[26, 25]. In a retrovirus, the viral genome is coded in RNA. When the virus in-

vades a host cell, the viral RNA sequences are transcribed into DNA by a reverse transcriptase enzyme that the virus supplies. During this process, which is opposite to the central dogma, the RNA template is destroyed. Subsequently, the DNA form of the viral genome, *i. e.* the provirus, is incorporated into the host DNA. The host transcription mechanisms then reproduce the viral genome and necessary viral enzymes, along with its own.

In the example of the graph $Z_5 \oplus Z_2$ shown in Figure 1. The molecular form of this graph is shown in Figure 2. The idea is to represent vertices in the graph with duplex molecules, and edges, or generators, as single-stranded molecules. Each segment of the molecule is identical, producing a replication of the symmetry of the group in the molecular representation, and consists of non-hybridizing spacer regions and a hybridization region.

The design for construction of the molecular representation of the group is shown in Figure 2. The seed consists of a loop attached to a duplex. In addition, the first spacer regions, and the hybridization region for the first circular molecule are present. Hybridization is controlled by thermal cycling. Subsequently, an RNA template is hybridized to regions that overhang the initial hybridization. An additional RNA primer is hybridized halfway on the RNA template. The template has the complements of additional spacer, hybridization, and overhang regions. Reverse Transcriptase is added, and the primers are extended as DNA molecules. RNase is added which degrades the template and the RNA primer. At the end of this process, two free DNA ends of the molecule are available for the next hybridization of a circular molecule. The entire process can be repeated as often as desired, producing larger group representations. As a final step, the overhang regions can be degraded producing the finished graph.

6 Conclusion

We have identified a class of graphs that admit a protocol for self-assembly in DNA molecules. The protocol uses a variation of implementations of finite-state machines that have been shown in the lab to be feasible and reliable. An important problem in computing with molecules concerns errors. Errors detract from the efficiency of the reactions by effectively removing the molecule from participation in the computation in the intended way, and are likely to wreck the computation altogether. For self-assembly protocols, creating the appropriate regions of homology may not be sufficient to guarantee a complete assemblage of the target structure. The class of automatic graphs permits the use of recent methodologies that have been proven feasible to implement state machines to prevent possible errors and guarantee that the overall structure will have a high probability of completion.

There are a couple of other interesting features about Cayley graphs appropriate for computation with DNA. Cayley graphs have seen a number of many other applications in computer science (communication networks, see for example [16]). Cayley graphs can also be designed to encode computations of arbitrary degrees of complexity (See, for example, Garzon-Zalcstein [10]). Cayley graphs can be designed, in particular, as look up tables of instances of NP-complete problems, or even Turing unsolvable problems [10]. Admittedly, these graphs are not likely to be automatic. A generalization of the techniques employed here, however, is possible for building (segments of) more arbitrary Cayley graphs. If so, the methodology described in this paper can be used to look up answers in these graphs as an alternative way to performing computations the way they have been so far done in DNA based computing [5, 23].

Acknowledgments

Thanks go to Natasha Jonoska for a conversation with the first author on the idea of building computation universal graphs in DNA.

References

- [1] Adleman, L. M. (1994) Molecular Computation of Solutions to Combinatorial Problems. *Science* **266**, 1021–1024.
- [2] Adleman, L. M. (1994). On Constructing a Molecular Computer. In [19], 1-21.
- [3] L. Landweber, E. Baum (eds.) (1998), *DNA Based COmputers II*, Proc. 2nd Annual workshop, Princeton University, 1996. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. **44**. American Mathematical Society, Providence RI.
- [4] G. Baumslag (1994). Word Processing in Groups (A review of [9]). *Bull. Amer. Math. Soc.* **31**:1 (1994), 86-91.
- [5] D. Beaver (1996), A Universal Molecular Computer. In: [19] 29-36.
- [6] H.S.M. Coxeter, W.O. Moser: Generators and relations for discrete groups. Springer-Verlag, New York, 1972
- [7] R. Deaton, R.C. Murphy, M. Garzon, D.R. Franceschetti, S.E. Stevens, Jr. (1995). Good Encodings for DNA-based Solutions to Combinatorial Problems. In [3], 159-171.
- [8] R. Deaton, M. Garzon, R.C. Murphy, J.A. Rose, D.R. Franceschetti, S.E. Stevens, Jr.. (1998). On the Reliability and Efficiency of a DNA-Based Computation. *Phys. Rev. Lett.* **80**:2 (1998) 417-420.
- [9] David B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Paterson, and W. P. Thurston. *Word processing in groups*, Jones & Bartlett, 1992.

- [10] M. Garzon, Y. Zalcstein (1991). The complexity of Grigorchuk groups with application to Cryptography. *Theoretical Computer Science* **88**:1 (1991), 83-98.
- [11] M. Garzon, Y. Gao, J. Rose, R.C. Murphy, R. Deaton, D.R. Franceschetti, S.E. Stevens, Jr. (1998). In-Vitro Implementation of Finite-State Machines. Proc. 2nd Workshop on Implementing Automata, London, Ontario 1997. Lecture Notes in Computer Science **1436**, 56-74.
- [12] M.R. Garey, D.S. Johnson (1979). *Computers and Intractability*, Freeman, New York.
- [13] I. Grossman, W. Magnus: Groups and their graphs. Random House, New York, 1964
- [14] M. Hagiya, M. Arita, D. Kiga, K. Sakamoto, S. Yokohama, Towards parallel evaluation and learning of Boolean μ -formulas with molecules. In [17], 105-115.
- [15] N. Jonoska, S. Karl, M. Saito (1998). Three-dimensional Structures in Computing. In [22], 189-200.
- [16] S. Lakshmivarahan, J-S Jwo, S.K.Dhall: Symmetry in interconnection networks based on Cayley graphs of permutation groups: a survey. *Parallel Processing* **19** (1993) 361-407
- [17] J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba, R.L. Riolo, eds. (1997). *Proc. 2nd Annual Genetic Programming Conference*, Morgan Kaufmann.
- [18] J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba, R.L. Riolo, eds. (1998). *Proc. 3rd Annual Genetic Programming Conference*, Morgan Kaufmann.
- [19] R. Lipton, E. Baum, eds. (1996). *DNA Based Computers*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. **27**. American Mathematical Society, Providence RI.
- [20] W. Magnus, A. Karrass, D. Solitar: Combinatorial group theory. Dover, New York, 1975
- [21] H. Rubin. D. Wood (eds.), 3rd DIMACS workshop on DNA Computers. Proc. of the wrd workshop, University of Pennsylvania, 1997.
- [22] H. Rubin. D. Wood (eds.), 4th DIMACS workshop on DNA Computers. Proc. of the 4th workshop, University of Pennsylvania, 1998.
- [23] W. Smith (1996), DNA Computers in vitro and vivo. In: [19] 121-185.
- [24] N. Seeman et al. (1996). The Perils of Polynucleotides: The Experimental Gap between the Design and Assembly of Unusual DNA Structures. In [3], 191-205.
- [25] L. Stryer (1995). *Biochemistry* 4th ed. W. H. Freeman and Company, New York.
- [26] J. D. Watson, N. H. Hopkins, J. W. Roberts, J. A. Steitz, and A. M. Weiner (1987). **Molecular Biology of the Gene**. The Benjamin/Cummings Publishing Co., Inc, Menlo Park, CA fourth edition.
- [27] E. Winfree (1996), On the Computational Power of DNA Annealing and Ligation: In [3], 199-221.
- [28] E. Winfree, Whiplash PCR for $O(1)$ Computing, In [22], 175-188.
- [29] E. Winfree, (1998) Simulations of computing by self-assembly. In [22], 213-240.
- [30] E. Winfree. F. Liu, L.A. Wenzler, N.C. Seeman (1998). Design and Self-Assembly of Two-Dimensional DNA Crystals. *Nature* **394**, 539-544.