Exploring DNA Strand-Displacement Computational Elements

Luca Cardelli[1], Andrew Phillips[1], Simon Youssef [2]

[1]Microsoft Research Cambridge
[2]Ludwig Maximilians Universität Munich

Abstract

The design of DNA strand-displacement systems presents a number of logical pitfalls, even beyond all the care that needs to be taken while designing non-interfering DNA codings [5]. These logical pitfalls include unwanted interference between elements of a single construction, unwanted interference between separately designed constructions, and unimagined interactions due to the sheer combinatorial complexity of the systems, sometimes affecting performance rather than functionality. We believe that all these issues will eventually require formal verification, but here we carry out a more empirical analysis of the state space to verify whether certain systems function as expected, and to gain experience in their logical and functional design.

We investigate a number of relatively simple signal-processing constructions represented in a programming language, DSD [1], able to describe a class of DNA strand-displacement systems that are composed only of a finite number of species and reactions, meaning that we can compute all the species that arise during the execution of a system from its initial components. Even for small source programs, the set of species and reactions can easily grow to a large size, and hence it is useful to have a tool to analyze them.

Our Visual DSD system is such a tool: it compiles a collection of DNA molecules into a set of chemical reactions. It also includes a stochastic simulator which computes a possible trajectory of the system and graphs the populations of species over time, and an ODE simulator. We use Visual DSD to investigate implementations of some basic computational elements (‘*gates*’) that are sufficient to represent interesting classes of dynamical systems; namely chemical reaction networks [3], stochastic Petri nets, and interacting automata [4].

Gate structures are consumed during signal processing: the energy driving the reactions is in fact provided by the gate structures being turned into waste structures. Hence the gate population is not fixed, and the kinetics of signal processing changes over time. One could use a very large and hence almost-constant concentration of gates with respect to the concentration of signals, but this puts limits on the concentration of signals. Moreover, the instantaneous concentration of signals is dependent on the dynamics of the reactions, and it may be difficult in general to keep it at a relatively low level. Finally, if there is a very large amount of free toeholds (given by very large populations of active gates) then the signals may too often bind reversibly to the ‘wrong’ gates, impeding progress. The optimal situation would be to keep the concentration of active gates at a constant level that is close to the concentration of the signals.

To alleviate these problems, an automatic buffering technique is proposed abstractly in [4]: here we flesh it out with concrete DNA structures. The idea is to keep a quasi-constant but relatively low

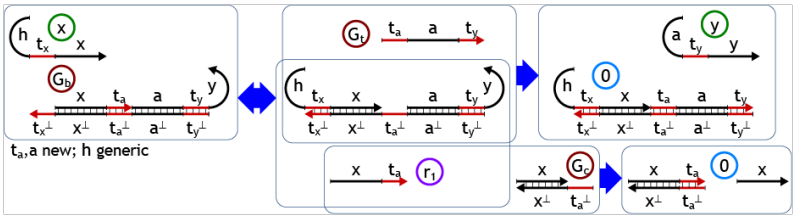

Figure 1: A Transducer and its Reactions

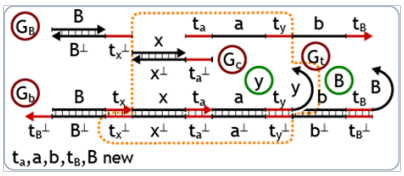

Figure 2: Buffered Transducer

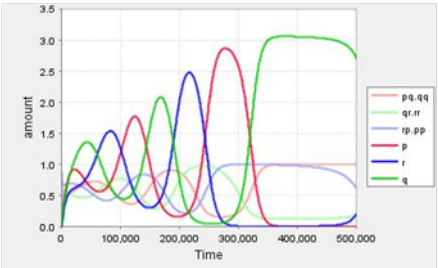

Figure 3: Buffered Oscillator Simulation

concentration of gate structures by means of a higher concentration of buffer structures that are turned into gates on demand. The buffer levels do not significantly affect the kinetics of the reactions (at least until they run out), and they could be replenished periodically without significantly affecting the on-going kinetics of the gates. The effective rates of the signal processing reactions remain then almost constant, provided the gates are replenished fast enough from the buffers.

Figure 1 shows the structures and reactions for a signal transducer converting strands representing a signal x to strands for a signal y. Figure 2 shows the buffered version of the transducer, where an additional signal B is used to activate gates from a buffer of inactive gates. Figure 3 shows the simulation of an oscillator composed of 3 buffered gates (*join* gates rather than transducer gates) [4].

In summary, we shown that automatic tools for compiling higher-level languages to (large sets of) chemical reactions can be useful for investigating DNA gate designs. Even when the set of reactions is finite, it can grow combinatorially with the size of the system, and it is unfeasible to generate it by hand. This is particularly important when including *unproductive* reversible reactions, which have a sometimes mild but usually noticeable effect on the kinetics, and other kinds of reactions like leaks and secondary structure interactions that we have not included in this work.

In some cases, simple finite sets of components can interact so intricately that it is not feasible to generate the full set of reactions; other systems (those that can produce polymers) have an infinite set of reactions. Still, all these systems can be described in high-level languages, because a language can represent a huge or even infinite state space (a set of species and reactions) finitely and compactly. Even when the state space cannot be precomputed, it is possible to inspect it by generating the required states incrementally from the high-level description. All this points to a useful role for high-level languages and tools for investigating DNA systems, which are by nature highly combinatorial.

References

[1] A. Phillips, L. Cardelli. A Programming Language for Composable DNA Circuits. Journal of the Royal Society Interface, August 6 2009, 6:S419-S436.
[2] A. Phillips, L. Cardelli. Efficient, Correct Simulation of Biological Processes in the Stochastic Pi-calculus. In Computational Methods in Systems Biology, LNBI 4695, pp 184-199, Springer, 2007.
[3] D. Soloveichik, G. Seelig, E. Winfree. DNA as a Universal Substrate for Chemical Kinetics. In Proc. DNA Computing and Molecular Programming, 14th International Conference, 2008.
[4] L. Cardelli. Strand Algebras for DNA Computing. In DNA Computing and Molecular Pro-gramming, Revised Selected Papers. LNCS 5877, pp 12-24, Springer, 2009.
[5] A. Marathe, A.E. Condon, R.M. Corn. On Combinatorial DNA Word Design. J. Comp. Biology 8(3) 201–219, 2001.