Molecular computations with competitive neural networks that exploit linear and nonlinear kinetics

Anthony J. Genot, Teruo Fujii, and Yannick Rondelez
LIMM/IIS-CNRS, University of Tokyo, Japan.
rondelez@iis.u-tokyo.ac,jp

Abstract

We show how to exploit enzymatic saturation -an ubiquitous nonlinear effects in biochemistry- in order to process information in molecular networks. The networks rely on the linearity of DNA strand displacement and the nonlinearity of enzymatic replication. We propose a pattern-recognition network that is compact and should be robust to leakage.

1 Introduction

Molecular programming aims to exploit chemical reactions and physical effects in order to process information at the molecular scale. Rather than solving complex mathematical problems (which is best left to electronic computers), molecular programming aims to provide robust and reliable methods to control and program chemical systems. For example, molecular computers would allow programmable chemical synthesis or smart therapeutics.

DNA has proved to be a versatile polymer to build and program at the nanoscale. The predictability of DNA assembly has allowed the construction of a rich library of systems: molecular transporters[1] , colloidal crystals [2] or a nanoscale map of the USA[3]. Adleman showed in 1994 that DNA can also solve computational problems such as the traveling salesman[4]. Since then, a variety of logic circuits have been demonstrated [5-7]. Qian, Winfree and Bruck have recently built neural networks entirely based on the mechanism of DNA strand displacement [8]. In this mechanism, a strand of DNA (the output) is selectively displaced from a logic gate by another strand of DNA (the input). Such gates are then assembled to build advanced neural networks such as associative memories or XOR functions.

Neural networks compute linear and nonlinear functions: summation, in which a neuron sums input signals, and activation, in which the sum is nonlinearly amplified and outputted by the neuron. Strand displacement is excellent to implement summations since it follows mass action kinetics, which is linear in each reactant [9, 10].

Strand displacement is leaky, which makes it difficult to cascade nonlinear computations because leakage is also cascaded. A leaky signal from one gate liberates other leaky signals from gates downstream, which amplifies exponentially leakages if the signals are catalytic or looped - a requirement for the activation of neurons.

Enzymatic reactions are much more specific and selective than DNA strand displacement. Enzymes have been used to actuate various molecular devices oscillators [11, 12] encoded in DNA sequences. It is well known that enzymatic reactions are prone to saturation, generally described with the Michaelis-Menten equation. Moreover, when two different substrates compete for the same enzyme, saturation will indirectly couple their kinetic: increase in the concentration of one substrate automatically decreases the processing rate of the other [13]. Far from being a nuisance, this phenomenon can produce a highly nonlinear effect known as winner-take-all, which digitally compares concentrations by amplifying infinitesimal differences[14] . The winner-take-all scheme has computational implications. Maass has shown that a single k-winner-take-all unit (which amplifies the k strongest signals) offers the same computational power as a multi-layers perceptron [15].

Here we propose neural networks that use the linearity of strand displacement for summation and the nonlinearity of enzymes for activation. A strand displacement layer computes several weighted sums of the inputs. This linear layer controls an enzymatic layer that absorbs all nonlinear computations through a winner-take-all effect. The enzymatic layer amplifies the maximal sum at the expense of the others.

Such winner-take-all networks should be more robust than circuits only based on strand displacement because they shift the burden of non-linearity to the enzymatic layer.

The chemical reactions used are presented thereafter. For illustration purpose, we base our demonstration on standard biochemical reactions classically used in molecular programming applications. We will also use reaction rates extracted from the corresponding reports. We note however that the presented mechanism is general and not restricted to this particular set of reactions.

The inputs X_i are digital: the initial concentration of the input strand is either 0 (FALSE) or c_0 (TRUE). The summation is based on the mass action kinetics of strand displacement. An input X_i displaces an inhibiting strand I_{ij} from a weight complex $W_{ij} = I_{ij}{:}T_j$ to yield an activated template T_j

The mass action constant k_f is on the order of $10^6/M/s$, which gives a half-time on the order of 10 seconds for initial concentrations of inputs and weights at 100 nM. In this case, strand displacement give rise to an irreversible reaction. A summation of concentrations naturally appears when different inputs activate the same template (fan-in). The concentration of activated template T_j after completion is $T_j^\infty = \sum_{i=1}^n \min(W_{ij}^0, X_i^0)$. Because X_i^0 is either 0 or c_0, and c_0 is in large excess over the weights, T_j^∞ simplifies to the weighted sum

$$T_j^\infty = \frac{1}{c_0}\sum_{i=1}^{n} W_{ij}^0 X_i^0$$

In turn, an activated template T_j catalyzes the replication of an output Y_j . This autocatalysis proceeds as follows: the template binds to Y_j to form a primer-template substrate, which triggers the polymerization of the 3' end of Y_j by a polymerase. A nicking enzyme recognizes this polymerization and nicks the elongated Y_j to yield two output strands Y_j and a free template.

The input and output strands are continuously degraded by an exonuclease that recognizes single-stranded DNA. The templates are chemically protected from degradation. The net reactions for Y j are:

$$Y_j + T_j \rightarrow 2Y_j + T_j$$

And

$$Y_j \rightarrow \emptyset$$

The kinetics of Y_j are summarized by the following equation:

$$Y_j' = \frac{\alpha T_j Y_j}{K_m K_b + \sum_{i=1}^{n} T_i Y_i} - k_r Y_j \ (1)$$

The first term corresponds to the polymerase-mediated replication of Y_j following Michaelis-Menten kinetics. The replication kinetics are nonlinear: the rate of replication is proportional to

$T_j Y_j$ for small concentrations, but for large concentrations the polymerase saturates and the rate becomes constant. The sum on the *n* outputs appears in the denominator because the polymerase is a resource shared by all outputs. This sum couples the kinetics of all outputs and generates the winner-take-all effect. The second term corresponds to the degradation of Y_j by the exonuclease, for which we assume pseudo first-order kinetics.

Winner-take-all is a nonlinear effect that occurs as soon as several outputs compete for the polymerase. As can be seen in (1), an increase in the concentration of one output Y_j accelerates its replication while it decelerates the replication of all other outputs. Small differences in concentration therefore snowball to such extent that, at the steady state, the fastest replicating output monopolizes the polymerase and bars access to the replication machinery to other outputs. Those losing outputs disappear due to their continuous degradation by the exonuclease.

We therefore have a two steps process. Strand displacement computes several weighted sums of the inputs. A winner-take-all based on a saturated enzymatic amplification mechanism digitalizes these results by picking the maximal sum and extinguishing the others. The surviving output Y_j corresponds to the template with the maximal concentration, i.e the template that maximizes the weighted sum $T_j^{\infty} = \frac{1}{c_0}\sum_{i=1}^{n} W_{ij}^{0} X_i^{0}$.

Simulation of a network that recognizes patterns is shown in Figure 1. Following Qian and colleagues [8], the network takes as input the answers to four questions about a scientist and returns as output the corresponding scientist. Input strands X_1 to X_4 encode answers to the questions. A bias input X_0, always set to c_0, encodes thresholds. The network's answer is given by the surviving output strand at the steady state. A variant of the perceptron algorithm optimized the weights, such that the questions X associated to a scientist Y_j maximize the weighted sum$\sum_{i=1}^{n} W_{ij}^{0} X_i^{0}$. The results are shown in Figure 1. The simulation shows that the half-time for the computation is about 5 minutes. For comparison, the half-time for similar computations with a DNA-only network ranges from 30 minutes to 10 hours [8].The network is compact because it requires only 22 strands. For comparison, a similar computation entirely based on DNA strand displacement would require about 120 strands [8]. We can see two factors that contribute to this compactness. Firstly DNA strands encode the network and enzymes are in charge of the digitization machinery. In DNA-only networks, digitalization is done by a DNA machinery specific to each gate. Secondly, competitive inhibition between *n* outputs usually requires $O(n^2)$ connections between them. We do not need those mutual connections because the winner-take-all is a global effect: the variation of one output immediately affects the replication rate of all others.

In summary, we have proposed neural networks that exploit the inherent strengths of linear and nonlinear kinetics. This combination should enable compact, quick and robust neural networks.

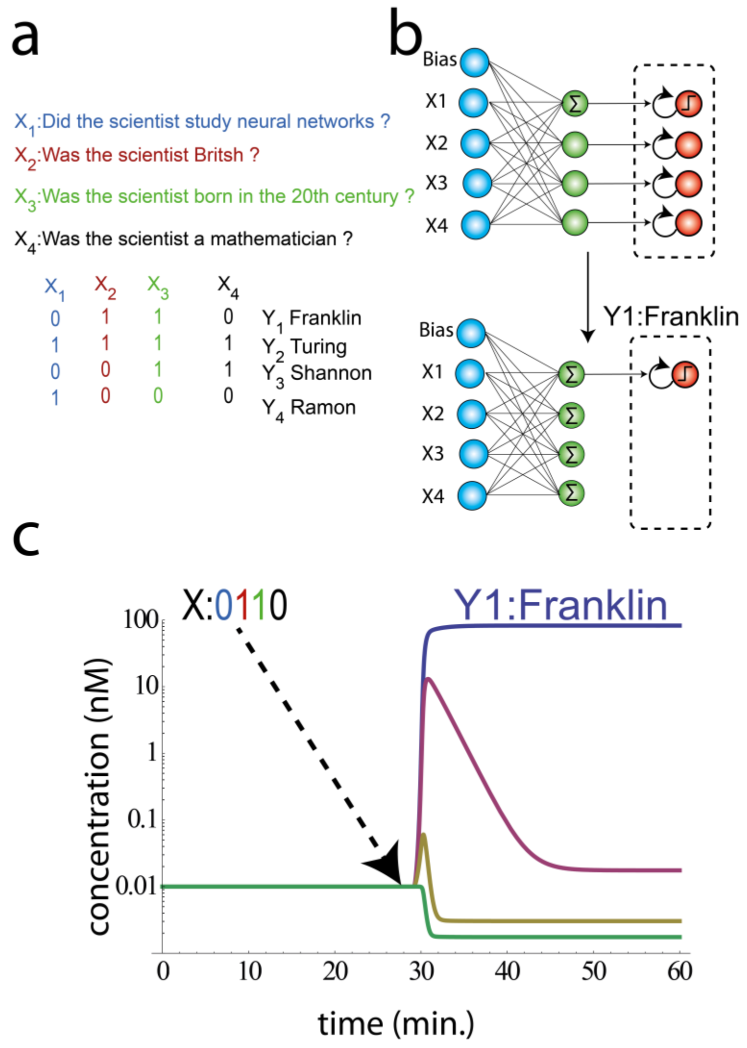

Figure 1: Simulation of a pattern-recognition network. The network takes as input four questions about a scientist and returns the corresponding scientist. (a) Questions associated to scientists. (b) Organization of the network. The network's answer is given by the surviving output at the steady state (c) Simulated kinetic evolution of the network for a question. The inputs are injected at t=30 minutes.

1. Sherman, W. B.; Seeman, N. C., *Nano Letters* **2004,** *4* (7), 1203-1207.
2. Nykypanchuk, D.; Maye, M. M.; van der Lelie, D.; Gang, O., *Nature* **2008,** *451* (7178), 549-552.
3. Rothemund, P. W. K., *Nature* **2006,** *440* (7082), 297-302.
4. Adleman, L. M., *Science* **1994,** *266* (5187), 1021-1024.
5. Seelig, G.; Soloveichik, D.; Zhang, D. Y.; Winfree, E., *Science* **2006,** *314* (5805), 1585-1588.
6. Qian, L. L.; Winfree, E., *Science* **2011,** *332* (6034), 1196-1201.
7. Genot, A. J.; Bath, J.; Turberfield, A. J., *Journal of the American Chemical Society* **2011,** *133* (50), 20080-20083.

8. Qian, L. L.; Winfree, E.; Bruck, J., *Nature* **2011,** *475* (7356), 368-372.
9. Yurke, B.; Mills, A. P., *Genet. Progr. Evol. Mach.* **2003,** *4*, 111-122.
10. Zhang, D. Y.; Winfree, E., *Journal of the American Chemical Society* **2009,** *131* (47), 17303-17314.
11. Kim, J.; Winfree, E., *Molecular Systems Biology* **2011,** *7*.
12. Montagne, K.; Plasson, R.; Sakai, Y.; Fujii, T.; Rondelez, Y., *Molecular Systems Biology* **2011,** *7*.
13. Rondelez, Y., *Physical Review Letters* **2012,** *108* (1).
14. Kim, J.; Hopfield, J. J.; Winfree, E., **Neural Network Computation by *in vitro* Transcriptional Circuits**. In *Advances in Neural Information Processing Systems*, Saul, L. K.; Weiss, Y.; Bottou, L., Eds. MIT Press: 2004; Vol. 17, pp 681-688.
15. Maass, W., *Neural Computation* **2000,** *12* (11), 2519-2535.