Stochastic computing with biomolecular automata

Rivka Adar*[†], Yaakov Benenson*[†][‡], Gregory Linshiz*[‡], Amit Rosner[§], Naftali Tishby[§][¶], and Ehud Shapiro*[‡][||]

Departments of *Biological Chemistry and [‡]Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel; [§]School of Computer Science and Engineering and [¶]Center for Neural Computation, Hebrew University, Jerusalem 91904, Israel

Edited by Richard M. Karp, International Computer Science Institute, Berkeley, CA, and approved May 5, 2004 (received for review February 2, 2004)

Stochastic computing has a broad range of applications, yet electronic computers realize its basic step, stochastic choice between alternative computation paths, in a cumbersome way. Biomolecular computers use a different computational paradigm and hence afford novel designs. We constructed a stochastic molecular automaton in which stochastic choice is realized by means of competition between alternative biochemical pathways, and choice probabilities are programmed by the relative molar concentrations of the software molecules coding for the alternatives. Programmable and autonomous stochastic molecular automata have been shown to perform direct analysis of disease-related molecular indicators *in vitro* and may have the potential to provide *in situ* medical diagnosis and cure.

Stochastic computing has a broad range of engineering and scientific applications (1–3), in particular, the analysis of biological information (4–7). The core recurring step of a stochastic computation is the choice between several alternative computation paths, each with a prescribed probability. Digital electronic computers realize stochastic choice in a costly and indirect way, through a software routine that invokes a pseudorandom number generator and analyzes the result to determine which alternative to choose (8). Although analog electronic devices for generating random numbers were proposed (9), they have not found their way into the mainstream. Furthermore, speeding up the generation of random numbers does not alleviate the need to process each of them to perform the stochastic choice. Biomolecular computers (10–13) use a different computational paradigm from electronic computers and hence may offer a radically different approach to this fundamental computing task.

Biomolecular computers are molecular-scale, programmable, autonomous computing machines in which the input, output, software, and hardware are made of biological molecules (12). Biomolecular computers hold the promise of direct computational analysis of biological information in its native biomolecular form, eschewing its conversion into an electronic representation. Recently this capability was shown to afford direct recognition and analysis of molecular disease indicators, providing *in vitro* disease diagnosis, which in turn was coupled to the programmed release of the biologically active molecule modeled after an antisense DNA drug (13). Because of the stochastic nature of biomolecular systems (14), a stochastic biomolecular computer would be more suitable for this biomedical task than a deterministic one.

Although the vision of a universal biomolecular computer was proposed three decades ago (15, 16), experimental research in this area began only a decade ago (17). Initially, research focused on large-scale DNA computers in which a human operator, or a large-scale robot, uses parallel DNA manipulation operations to achieve high performance in solving computationally intensive problems (18–25). A different research track demonstrated molecular systems that go through a predetermined sequence of state-to-state transitions in a deterministic (26) or stochastic (27) fashion and programmable, autonomous computing machines with molecular input, output, software, and hardware that realize two-state, two-symbol finite automata (10–12). A mathematical description of such an automaton is shown in Fig. 1*A*, and an example computation is shown in Fig. 1*B*. In the molecular realization of these automata the input is encoded as a single DNA molecule, transition rules are encoded by another set of DNA molecules, and the hardware consists of DNA-manipulating enzymes. A computation commences when all molecular components are present in solution and processes the input molecule in steps performed by the hardware molecules, as directed by the software molecules. The output molecule, formed by the programmed digestion of the input molecule, encodes the result of the computation.

Although these experimental realizations demonstrate primarily deterministic computations, both the mathematical notion of automata (28) and its molecular realizations afford a stochastic extension by allowing multiple competing transitions to apply to any state-symbol combination. An automaton is said to be stochastic if each elementary transition is ascribed a certain probability (29), and the sum of probabilities of all transitions applicable to a given state-symbol combination is 1. A stochastic finite automaton is a simple notional computing machine. It is compared with a deterministic finite automaton in Fig. 1. Unlike a deterministic automaton, which is programmed by selecting a specific set of transitions, at most one for each state-symbol combination, a stochastic automaton potentially uses all transition rules, ascribing each transition with a predefined probability. The output of a stochastic computation is the probability to obtain each final state, computed from the probabilities of single transitions by summing the probabilities of all possible computation paths that result in the same final state. Stochastic automata are useful for the analysis of sequences or processes that are not deterministic (3–5).

Here, we demonstrate a design principle for stochastic computers, afforded by the unique properties of biomolecular computers, to realize the intended probability of each transition by the relative molar concentration of the software molecule encoding that transition. We describe and experimentally analyze a stochastic molecular automaton that operates according to this principle. The experiments show robustness of programmed transition probabilities to input molecule concentrations and to absolute software molecule concentrations and a good fit between predicted and actual result probabilities of multistep computations.

Materials and Methods

Assembly of the Components. Software and input molecules: single-stranded synthetic oligonucleotides (desalted and lyophilized, 1-μmol scale; Sigma-Genosys) composing the software and input molecules were purified to homogeneity by using a 15% denaturing acrylamide gel (40 cm × 1.5 mm) containing 7 M urea. Input oligonucleotides used for the calibration of the different concentration ratios of the transition molecules and for constructing the long inputs were labeled with carboxyfluorescein (FAM) at the 3′ of the sense DNA strand and with CY5 at the 5′ of the antisense DNA strand (Sigma-Genosys). The names

This paper was submitted directly (Track II) to the PNAS office.

Freely available online through the PNAS open access option.

Abbreviation: FAM, carboxyfluorescein.

[†]R.A. and Y.B. contributed equally to this work.

[||]To whom correspondence should be addressed. E-mail: ehud.shapiro@weizmann.ac.il.

www.pnas.org/cgi/doi/10.1073/pnas.0400731101

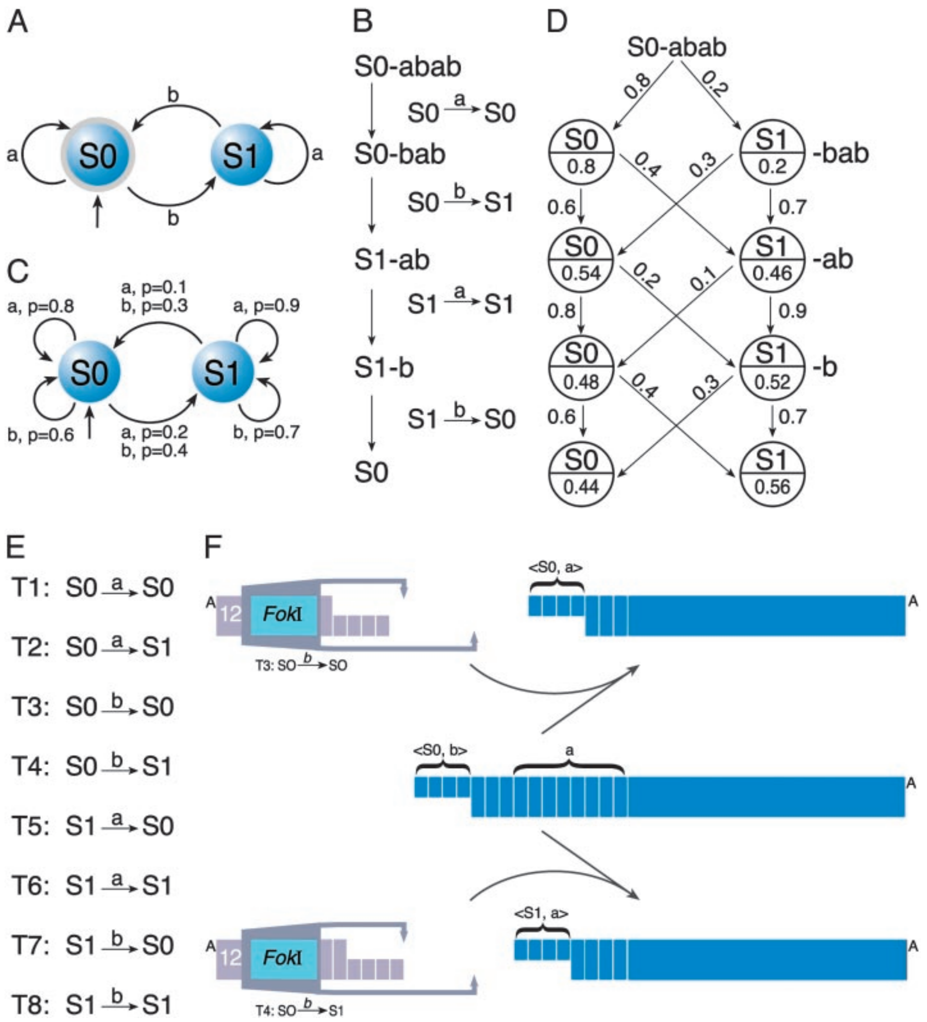

Fig. 1. Deterministic and stochastic finite automata. (*A*) Deterministic finite automaton. The automaton has two states, S0 and S1, and can process sequences containing the symbols *a* and *b*. The incoming unlabeled arrow marks the initial state and labeled arrows represent transition rules, each specifying the next state based on the current state and the current symbol. The diagram shows an automaton that determines whether an input string contains an even number of *b* symbols. (*B*) The linear computation path of the deterministic automaton processing the input *abab*, including the configurations (state-input combinations) that arise during the computation and the sole transition that applies to each configuration. (*C*) Stochastic finite automaton. This automaton differs from the deterministic one in that two competing transitions rather than one are applicable to each state-symbol combination. The probability of each transition to be chosen is indicated in the diagram. The output of the computation is a probability distribution over the final states rather than a single final state. (*D*) The computation graph of the stochastic automaton processing the input *abab*, including probabilities of choosing each transition and probability distributions of intermediate configurations and final states. (*E*) Software. The complete list of transition rules of the two-state two-symbol automaton shown in *B*. (*F*) Competing biochemical pathways of the stochastic molecular automaton on a configuration in which the state-symbol combination is ⟨S0, *b*⟩, and the two applicable transitions are T3 and T4.

and sequences of the oligonucleotides were: A.cal.s (CAGGGCCGCAGGGCCGCAGGGCCTGGCTGCCAAAAATTACCGATTAAGTTGGA-FAM), A.cal.as (Cy5-CCAACTTAATCGGTAATTTTTGGCAGCCAGGCCCTGCGGCCCTGCGGC), B.cal.s (GGCTGCCTGGCGGCCTGGCTGCCGCAGGGCCAAAAATTACCGATTAGTTGGA-FAM), B.cal.as (Cy5-CCAACTTAATCGGTAATTTTTGGCCCTGCGGCAGCCAGGCCGCCAGGC). The input *bbba* was prepared by the annealing of A.cal.s and A.cal.as oligonucleotides, and the input *aaab* was prepared by the annealing of B.cal.s and B.cal.as oligonucleotides. The construction of the long inputs *abbbbbba*, *babbaaab*, *baaaaaab*, *babbabbbbbba*, *baaaabbbbbba*, *abbbbabbaaab*, *abbbbaaaaaab*, and the T1–T8 software molecules was described elsewhere (11).

Calibration Reactions. Calibration of the different concentration ratios of the software transition pairs was performed by using 0.1 μM concentrations of four-symbol inputs. In a typical reaction, a program required for a particular calibration (Fig. 2*A*) was composed as follows: for each deterministic step, a 0.5 μM concentration of the corresponding transition was added. Thus, the deterministic part of the computation was performed by a total of 1.5 μM transition mixture. For the last-choice step, a total of the 0.5 μM concentration of the tested transition molecules mixture was taken at a required ratio. *Fok*I enzyme was added at the 2 μM final concentration to maintain the stoichiometric ratio with the software molecules. Before input addition reaction mixtures were preincubated with *FokI* at 15°C for 20 min. The reactions were quenched after 2 h and run on denaturing acrylamide gel (40 cm × 0.4 mm, 15%, 7 M urea) with low-fluorescence plates. The fluorescence was read by using the TYPHOON SCANNER CONTROL software of the Typhoon 9400 machine and quantified by using the IMAGEQUANT V. 5.2 software (Amersham Pharmacia Biosciences). The quantitation was per-

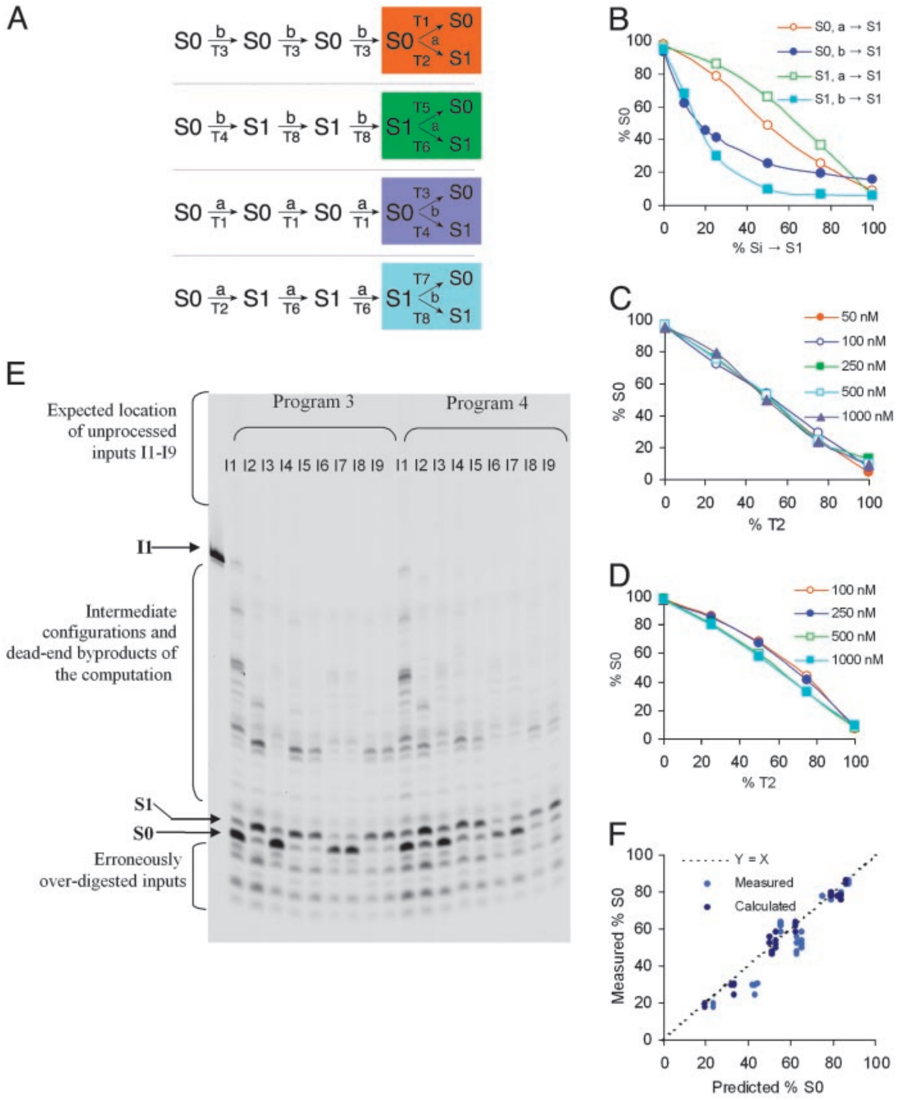

Fig. 2. Experimental support to stochastic computation reactions. (*A*) Computation trees used for the calibration of transition probabilities. (*B*) The mapping from the relative concentration of the competing transition molecules to the distribution of the output states for a single stochastic choice. Each curve shows the result for a different pair of competing transitions. Each pair is represented in the legend by a transition rule transforming to the S1 state. (*C*) Sensitivity of the probability distribution to the absolute concentration of the input molecule. Distribution of the output states for a single stochastic choice is shown for different absolute input concentrations and different ratios of the transition molecules. The computation was performed on the input *bbba* by using the transition T3 ($S0 \xrightarrow{b} S0$) for the first three symbols and a mixture of the transitions T1 ($S0 \xrightarrow{a} S0$) and T2 ($S0 \xrightarrow{a} S1$) for the choice step. The deterministic software molecule T3 was kept at 1.5 μM concentration and the choice molecules T1 and T2 were maintained at a total concentration of 0.5 μM, while their ratio was systematically varied. Input concentrations were 50, 100, 250, 500, and 1,000 nM. For each input concentration and ratio of the competing transition molecules, the output ratio was measured. Each curve summarizes the results for one input concentration, indicated in the legend. (*D*) Sensitivity of the probability distribution to the absolute concentration of the software molecules. Input concentration was kept constant, whereas the total concentration of the competing transition molecules was 100, 250, 500, and 1,000 nM. Distribution of the output states for a single stochastic choice is shown for different total concentrations and different ratios of the software molecules. Each curve summarizes the results for one total concentration of the software molecules, indicated in the legend. The computation used was the same as for the results shown in *C*. (*E*) Experimental data used for measuring the output distribution. The gel shows the results obtained with programs 3 and 4 (Table 1). (*F*) The correlation between the predicted and the measured output distribution for all computations by using measured and calculated calibrations.

formed according to the Cy5 label (PMT 500–550 V) on the 5′ terminus of the antisense strand, because it gave more reproducible and consistent results than the FAM label on the 3′ terminus of the sense strand. Excitation was done with the red laser (633 nm), and emission was measured through the 670 BP30 filter.

Computation Reactions. In a typical reaction, the ratios of the input, the software, and the hardware were 0.1:2:2 (input/transition molecules/*Fok*I, respectively). Each pair of competing transition molecules was maintained at 0.5 μM concentration. Computation reactions containing the input, transition molecules, and the *Fok*I class IIS restriction enzyme (54 μM stock; 60 units/μl; New England Biolabs) were performed in 10 μl of NEB4 buffer at 15°C. Before input addition reaction mixtures were preincubated with *FokI* at 15°C for 20 min. After 2 h, 2-μl aliquots were taken from the reaction mixtures, added to 4 μl of stop solution [9 volumes of Formamide (Merck) and 1 volume of 10× 89 mM Tris/89 mM boric acid/2 mM EDTA, pH 8.3 (TBE)]. Half of the sample above was assayed by 15% denaturing

PAGE containing 7 M urea (ultrapure, ICN). The bands were visualized by the Typhoon 9400 machine. In this assay, formation of S0 or S1 outputs was represented by 16- or 17-nt-long product bands with the Cy5 label. Quantitation of the products was done by the IMAGEQUANT V. 5.2 as described in *Calibration Reactions*.

Calculation of Transition Probabilities. To calculate the transition probabilities by using measured output distribution, a computer implementation of the computation graph was developed in MATLAB 6.5 (MathWorks, Natick, MA). The simulation generated an equation set for each given program, with transition probabilities as the unknown variables. Each equation summed the probabilities of all possible computation paths carried out on one input. The result was expected to be similar to the measured final-state distribution for that computation. A solution to such an equation set is an optimal set of transition probabilities, which minimizes the discrepancy between the calculated and the measured final-state distribution. A least-squares optimization was used for calculating the optimal solution by iteratively refining the transition probabilities by using the method of preconditioned conjugate gradients. However, because of the nature of the problem, there could be local solutions, which might not be consistent with other programs. The following process was used for finding a consistent solution: Programs 1, 2, and 3 were used jointly as training programs according to which the simulation could learn a consistent set of transition probabilities. The optimization process was repeated 450 times for programs 1, 2, and 3. For each set, a single optimization was performed given the calibration initial values. Additional 449 optimizations were performed for each program with random initial values. All possible combinations of the calculated optimal transition sets were compared with each other. The most consistent triplet-of-transition probability set was selected, so that the transition probabilities calculated for the same concentration ratios were similar for different programs.

Determining the Deviation of Predicted Results. Determination of the standard deviation of the predicted output ratio was performed by simulating all possible independent pipetting errors of 5% with the same probabilities when preparing the mixtures of the transition molecules. We simulated the experimental setup where each pair of transitions was mixed independently, and then the program mixture was composed from equal solution volumes containing each pair. We assumed discrete deviations of -5%, 0%, and 5% from the nominal volume of each software molecule solution. The deviation in the transition probability for each volume deviation was calculated from the measured calibration curve. Each program generated a set of 6,561 (3^8) different probability combinations, resulting in 6,561 possible output ratios, for which a standard deviation was calculated. The average of the set was very close to the predicted value with no deviations. The simulation was performed with the MATLAB 6.5 software (MathWorks).

Results

The molecular stochastic automaton described here is based on the two-state, two-symbol automaton developed in our laboratory (11). The hardware consists of a class IIS restriction enzyme *Fok*I, which recognizes the sequence GGATG and cleaves 9 and 13 nucleotides away from the recognition site on the sense and antisense strands, respectively. The automaton has eight possible transition rules (Fig. 1*E*). As transition molecules used in a computation are reusable, their concentration remains constant. This ensures that transition probabilities derived from transition molecule concentrations remain constant during the computation.

At the molecular level, two transition molecule species compete for any intermediate configuration with probabilities of success correlated to their relative concentrations (Fig. 1*F*). When the computation is performed on a large ensemble of input molecules, the probability to obtain a particular final state can be measured directly from the relative concentration of the output molecule encoding this state among all output molecules. We control the transition probabilities by varying the relative concentration of the transition molecules and attempt to predict the distribution between output molecules for a given input based on these concentrations. Our key problem in doing so is to determine the function linking relative concentrations of competing transition molecules to the probability of a transition being chosen. Minute variations in the chemical composition of each transition molecule result in different affinities to the input molecule and, therefore, this function must be determined experimentally. Furthermore, we cannot assume that this function is independent of the absolute molar concentrations of the input and/or of the software molecules. The bulk of our experimental and computational work was devoted to determining this function for each pair of competing transition molecules and to establishing its independence of these other parameters.

To determine the function mapping relative concentrations of transition molecules to transition probabilities we performed calibration with the four-symbol inputs *aaab* and *bbba*. The computation proceeds deterministically until the last symbol and then performs a stochastic choice between two competing transitions. We used this design to represent adequately multisymbol computations and avoid effects unique to initial symbols. We prepared software mixtures that represent all four possible intermediate state-symbol combinations (Fig. 2*A*). The computation was carried to completion and the exact duration was determined in the preliminary experiments with inputs of different lengths. The resulting measured calibration curves for all four competing pairs of transition molecules are shown in Fig. 2*B*, demonstrating a different mapping for each pair of transitions. Transitions processing the symbol *a* provide a linear mapping, with a probability distribution being close to the concentration ratio. On the other hand, transitions processing *b* reveal a convex mapping, which is apparently caused by the software molecules that result in state S1 (T4 and T8) having a higher reaction rate than the competing software molecules that result in state S0 (T3 and T7). In steep regions of the curve the precision with which probabilities can be programmed is reduced because of higher sensitivity to pipetting errors. Error probabilities of the software molecules are reflected in the non-zero probability to accomplish a transition when its competing transition is absent (e.g., T3–T4). However, we suspect that the computation reaction also has byproducts that overlap with error-representing bands but do not accumulate in the final error of multistep computations. Studies to distinguish between the two types of erroneous products are underway. Accumulating errors in transitions result from the incorrect cleavage by *Fok*I that also cleaves one nucleotide further than expected both in the sense and antisense strands of the input molecule. Because of this imprecision in enzyme action and the relative location of the S1 and S0 sticky ends, transitions that transform to state S1 may result in transitions to state S0, whereas transitions intended to transform to S0 result in dead-end products. This finding explains the higher apparent error rates observed with the software molecules that transform to S1.

To verify that the system is insensitive to fluctuations in input concentration, which naturally occur in the course of a multistep computation, we measured the distribution of output states of a computation with one stochastic choice by using the input *bbba* and the computation tree used for calibration of the competing pair of transition pair T1–T2. The summary of the results in Fig. 2*C* indicates that the computation is insensitive to the different input concentrations used and the transition probability is determined solely by the ratio between the transition molecules.

Table 1. Programs tested with the stochastic automaton

	Program 1			Program 2			Program 3			Program 4		
T	$[T]_{rel}$	P_m	P_c	$[T]_{rel}$	P_m	P_c	$[T]_{rel}$	P_m	P_c	$[T]_{rel}$	P_m	P_c
T2	50	51	43	20	18	27	20	18	27	50	51	43
T4	10	38	54	10	38	54	50	74	77	10	38	54
T6	50	34	33	20	12	13	20	12	13	20	12	13
T8	10	32	35	10	32	35	50	90	94	50	90	94

All values stated are percentages.

Another set of experiments was performed to ensure that the transition probability as determined by the ratio between transition molecules is not affected by their absolute concentrations. We used the same experimental system as above, but in this case the concentration of the input was kept constant while the total concentration of the competing transition molecules varied. The summary of the results is shown on the graph in Fig. 2*D*. The results indicate that the transition probability is indeed relatively insensitive to the absolute software concentrations and is defined mostly by the relative concentration ratio.

We tested the stochastic finite automaton by running four programs with the same structure as the one described in Fig. 1*C*, but with different transition probabilities (Table 1), on nine inputs that varied in symbol composition and length (I1–I9, Table 2, which is published as supporting information on the PNAS web site). Each program specifies the relative concentrations of all pairs of competing transition, which in turn determine their expected probabilities as explained below. We carried out extensive preliminary experiments (not shown) to develop the reaction protocol. These experiments indicated that at high concentrations the transition T6 probably digests other transition molecules with the aid of the enzyme *Fok*I, which is present in the solution, modifying the software composition during the computation in an unpredictable way. Hence, we avoided high concentrations of T6 in our programs. We then performed the calibration and computation experiments reported here in a single run under uniform conditions. Fig. 2*E* shows representative results of the computations as analyzed by PAGE. Each lane is an application of programs 3 and 4 to one of the inputs. We predicted the distribution of the outputs by using the calibration graphs (Fig. 2*B*). Good correlation was observed between predicted and measured results by using measured transition probabilities, although some systematic errors were apparent. To account for these discrepancies, we calculated the expected deviation in output probabilities caused by independent pipetting errors of 5% when mixing transition molecules (Fig. 3). The calculation indicated that a number of measured results, notably with inputs ending with *b*, fell outside of the expected error range and were consistently lower than the prediction. Therefore, this bias could not be attributed solely to pipetting errors but rather to some error in the method of direct probability measurement. To compensate for discrepancies between transition probabilities obtained in direct measurements and those manifested in multistep computations, we designed an

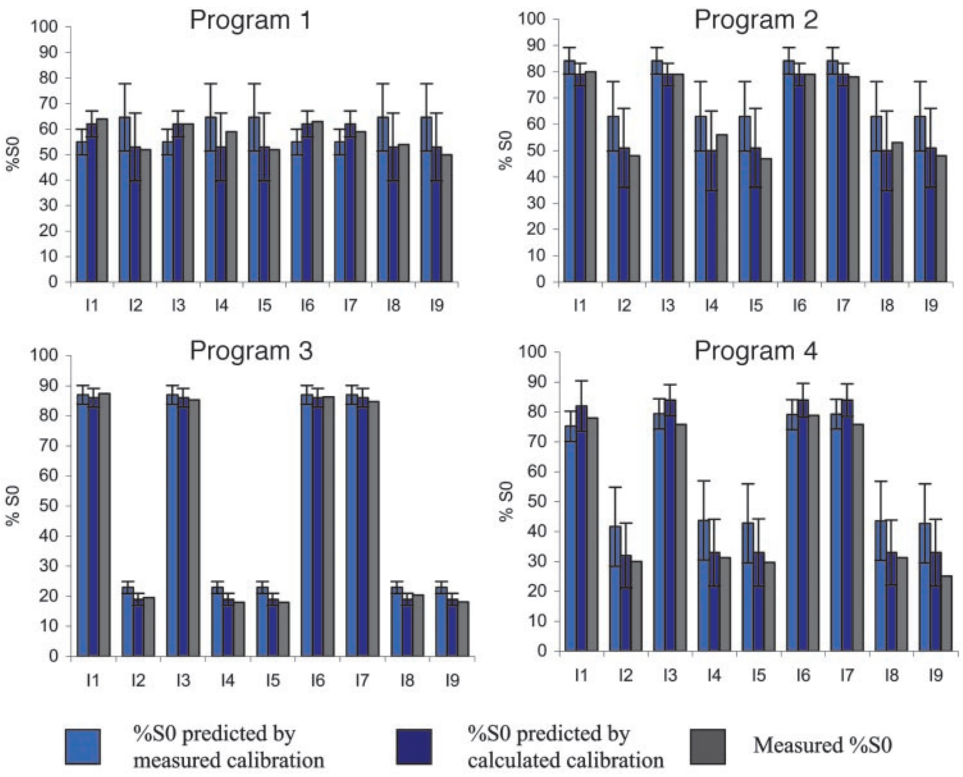

Fig. 3. Bar charts show predicted vs. measured S0 output probability for each experiment. Output probabilities are predicted by using measured calibration (light blue) and calculated calibration (dark blue) and compared with measured probabilities (gray). Error bars show ± 2 SD (95% of expected results) in predicted outputs caused by independent pipetting errors of 5%.

in silico method for probability determination based on a simulation of the reaction network. The simulation utilizes least-squares optimization to find an optimal set of transition probabilities for each program. The optimization begins with measured transition probabilities and iteratively refines them until a minimum discrepancy between calculated and measured output probabilities is reached. Programs 1, 2, and 3 were used jointly as a training set according to which the simulation could learn a consistent set of optimal transition probabilities. The transition probabilities calculated for the same concentration ratios were equal for different programs, as they should be under our molecular computational model. The computed transition probabilities were then tested independently on program 4, since it is composed of pairs of transitions already used in at least one of the programs 1–3, and it provided a good fit to the measured final-state probabilities of that program.

Some errors in direct probability measurements become apparent by comparing measured and calculated transition probabilities. The pair T3–T4, for example, has a measured probability of 38% to transform to S1 compared with a calculated probability of 54%. This finding suggests that the method for direct probability measurements should be improved in the future. In addition, a strong correlation exists between the standard deviation of predicted output probability and the difference between measured and predicted output probabilities, which indicates that for some transition pairs the differences resulting from intrinsic sensitivity to pipetting errors overlap with a systematic error in probability measurements.

In Table 1, column $[T]_{rel}$ shows the relative percentage of the transition molecules that transform to S1 within each competing pair; column P_m shows the measured transition probability corresponding to this molecule, and column P_c shows the calculated transition probability. Fig. 2*F* shows the correlation between measured output probabilities and output probabilities calculated from measured and calculated transition probabilities. The correlation that utilizes the calculated transition probabilities is very good. A detailed summary of the results is given in Tables 2 and 3, which are published as supporting information on the PNAS web site.

Discussion

Predictability and error control are prerequisites for any practical computer architecture. We obtained a good fit between predicted and measured computation output using calculated transition probabilities. This result suggests that the transition probability associated with a given relative concentration of a software molecule is a dependable programming tool. This principle was recently used in a construction of a molecular computer capable of probabilistic logical analysis of the disease-related molecular indicators *in vitro* by coregulating the concentration of software molecules with the concentration of these indicators (13).

We thank A. Regev for critical review of this manuscript. This work was supported by grants from the Israeli Ministry of Science, the Israeli Science Foundation, and the Minerva Foundation.

1. Maass, W. & Orponen, P. (1998) *Neural Comput.* **10,** 1071–1095.
2. Segala, R. (1995) *Lect. Notes Comput. Sci.* **962,** 234–248.
3. Delgado, J. & Sole, R. V. (2000) *Phys. Lett. A* **270,** 314–319.
4. Bejerano, G. & Yona, G. (2001) *Bioinformatics* **17,** 23–43.
5. Durbin, R., Eddy, S. R., Krogh, A. & Mitchison, G. (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids* (Cambridge Univ. Press, Cambridge, U.K.).
6. Priami, C. Regev, A., Shapiro, E. & Silverman, W. (2001) *Inf. Process. Lett.* **80,** 25–31.
7. Regev, A. & Shapiro, E. (2002) *Nature* **419,** 343–343.
8. Gentle, J. E. (1998) *Random Number Generation and Monte Carlo Methods* (Springer, New York).
9. Cauwenberghs, G. (1999) *IEEE Trans. Circuits II* **46,** 240–250.
10. Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z. & Shapiro, E. (2001) *Nature* **414,** 430–434.
11. Benenson, Y., Adar, R., Paz-Elizur, T., Livneh, Z. & Shapiro, E. (2003) *Proc. Natl. Acad. Sci. USA* **100,** 2191–2196.
12. Benenson, Y. & Shapiro, E. (2004) in *Dekker Encyclopedia of Nanoscience and Nanotechnology*, eds. Schwarz, J. A., Contescu, C. I. & Putyera, K. (Dekker, New York), pp. 2043–2056.
13. Benenson, Y., Gil, B., Ben-Dor, U., Adar, R. & Shapiro, E. (2004) *Nature* **429,** 423–429.
14. McAdams, H. H. & Arkin, A. (1997) *Proc. Natl. Acad. Sci. USA* **94,** 814–819.
15. Bennett, C. H. (1979) *BioSystems* **11,** 85–90.
16. Bennett, C. H. (1982) *Int. J. Theor. Phys.* **21,** 905–940.
17. Adelman, L. M. (1994) *Science* **266,** 1021–1024.
18. Lipton, R. J. (1995) *Science* **268,** 542–545.
19. Ouyang, Q., Kaplan, P. D., Liu, S. & Libchaber, A. (1997) *Science* **278,** 446–449.
20. Khodor, J. & Gifford, D. K. (1999) *Biosystems* **52,** 93–97.
21. Ruben, A. J. & Landweber, L. F. (2000) *Nat. Rev. Mol. Cell. Biol.* **1,** 69–72.
22. Sakamoto, K., Gouzu, H., Komiya, K., Kiga, D., Yokoyama, S., Yokomori, T. & Hagiya, M. (2000) *Science* **288,** 1223–1226.
23. Faulhammer, D., Cukras, A. R., Lipton, R. J. & Landweber, L. F. (2000) *Proc. Natl. Acad. Sci. USA* **97,** 1385–1389.
24. Mao, C., LaBean, T. H., Reif, J. H. & Seeman, N. C. (2000) *Nature* **407,** 493–496.
25. Stojanovic, M. N. & Stefanovic, D. (2003) *Nat. Biotech.* **21,** 1069–1074.
26. Sakamoto, K., Kiga, D., Komiya, K., Gouzu, H., Yokoyama, S., Ikeda, S., Sugiyama, H. & Hagiya. M. (1999) *Biosystems* **52,** 81–91.
27. Bar-Ziv, R., Tlusty, T. & Libchaber, A. (2002) *Proc. Natl. Acad. Sci. USA* **99,** 11589–11592.
28. Hopcroft, J. E., Motwani, R. & Ullmann, J. D. (2000) *Introduction to Automata Theory, Languages and Computation* (Addison-Wesley, Boston), 2nd Ed.
29. Rabin, M. O. (1963) *Inf. Control* **6,** 230–245.

BIOCHEMISTRY

COMPUTER SCIENCES