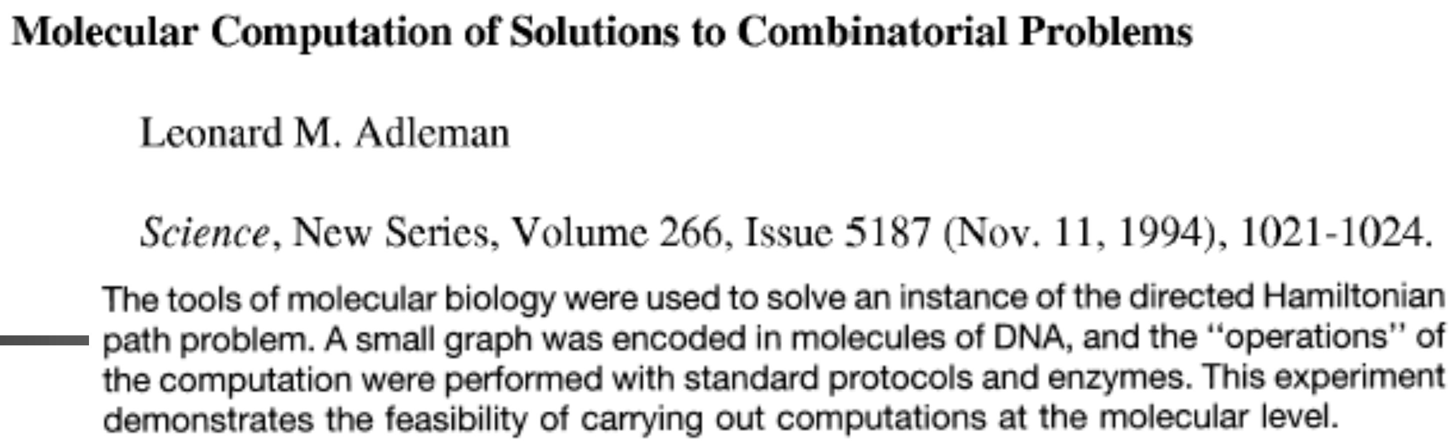
DNA-based Computing

- Adleman - Hamiltonian Path (7 node)
- Biochemical manipulation
- Sticker model, RNA, Whiplash PCR
- Surface-based computation
- Algorithmic self-assembly
- Autonomous systems



Molecular Computation of Solutions to Combinatorial Problems

Leonard M. Adleman

Science, New Series, Volume 266, Issue 5187 (Nov. 11, 1994), 1021-1024.

The tools of molecular biology were used to solve an instance of the directed Hamiltonian path problem. A small graph was encoded in molecules of DNA, and the "operations" of the computation were performed with standard protocols and enzymes. This experiment demonstrates the feasibility of carrying out computations at the molecular level.

- Hamiltonian path problem (HPP). Is there a path through the graph which visits each node exactly once? [Relation to TSP].
- NP problem (requires non-polynomial time to solve). Solution space increases exponentially as problem size increases linearly.
  - Hard to solve. Easy to check.
  - Oracle. Heuristic.
- Combinatorial optimization problems.

Molecular Computation of Solutions to Combinatorial Problems

Leonard M. Adleman

Science, New Series, Volume 266, Issue 5187 (Nov. 11, 1994), 1021-1024.

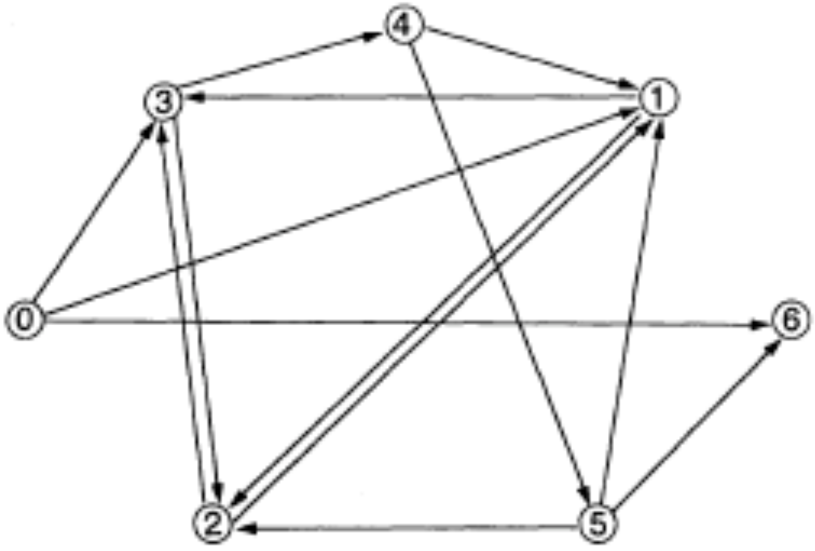
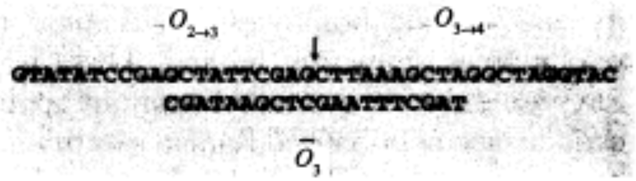


Fig. 1. Directed graph. When $v_{in} = 0$ and $v_{out} = 6$, a unique Hamiltonian path exists: 0→1, 1→2, 2→3, 3→4, 4→5, 5→6.

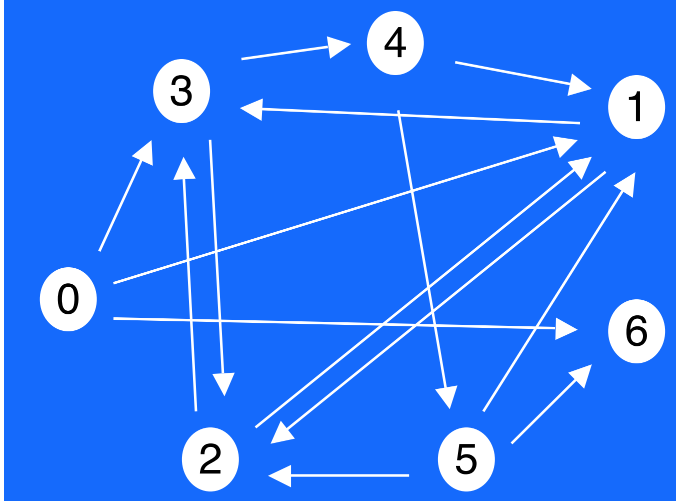
O_2	TATCGGATCGGTATATCCGA
O_3	GCTATTCGAGCTTAAAGCTA
O_4	GGCTAGGTACCAGCATGCTT
$O_{2\rightarrow 3}$	GTATATCCGAGCTATTCGAG
$O_{3\rightarrow 4}$	CTTAAAGCTAGGCTAGGTAC
$\bar{O}_3$	CGATAAGCTCGAATTTCGAT



GTATATCCGAGCTATTCGAGCTTAAAGCTAGGCTAGGTAC
CGATAAGCTCGAATTTCGAT

Fig. 2. Encoding a graph in DNA. For each vertex *i* in the graph, a random 20-mer oligonucleotide O_i is generated (shown are O_2, O_3, and O_4, for vertices 2, 3, and 4, respectively). For edge *i*→*j* in the graph, an oligonucleotide $O_{i\rightarrow j}$ is derived from the 3′ 10-mer of O_i and from the 5′ 10-mer of O_j (shown are $O_{2\rightarrow 3}$ for edge 2→3 and $O_{3\rightarrow 4}$ for edge 3→4). For each vertex *i* in the graph, $\bar{O}_i$ is the Watson-Crick complement of O_i (shown is $\bar{O}_3$, the complement of O_3). $\bar{O}_3$ serves as a splint to bind $O_{2\rightarrow 3}$ and $O_{3\rightarrow 4}$ in preparation for ligation. All oligonucleotides are written 5′ to 3′, except $\bar{O}_3$.

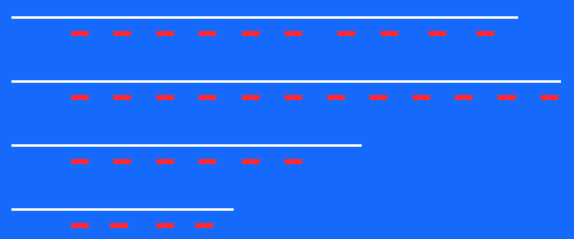
Adleman’s Experiment



O_3
GCTATTCGAGCTTAAAGCTA
$O_{2\text{->}3}$
GTATATCCGAGCTATTCGAG
$O_{3\text{->}4}$
CTTAAAGCTAGGCTAGGTAC

$O_{2\text{->}3}$ $O_{3\text{->}4}$
GTATATCCGAGCTATTCGAGCTTAAAGCTAGGCTAGGTAC
CGATAAGCTCGAATTTCGAT
$\sim O_3$

1). Generate all possible paths.
2). Separate and discard incorrect paths.



Science 266, 1021 (1994)

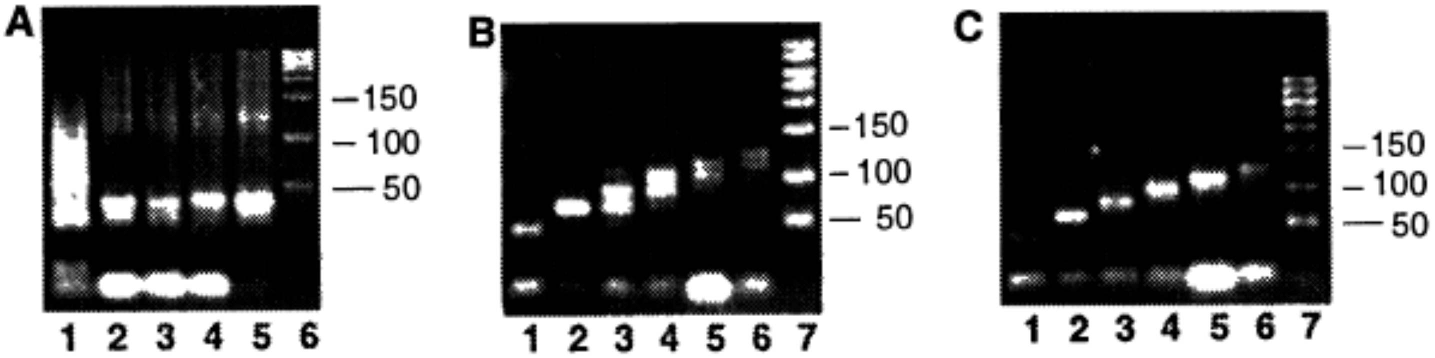
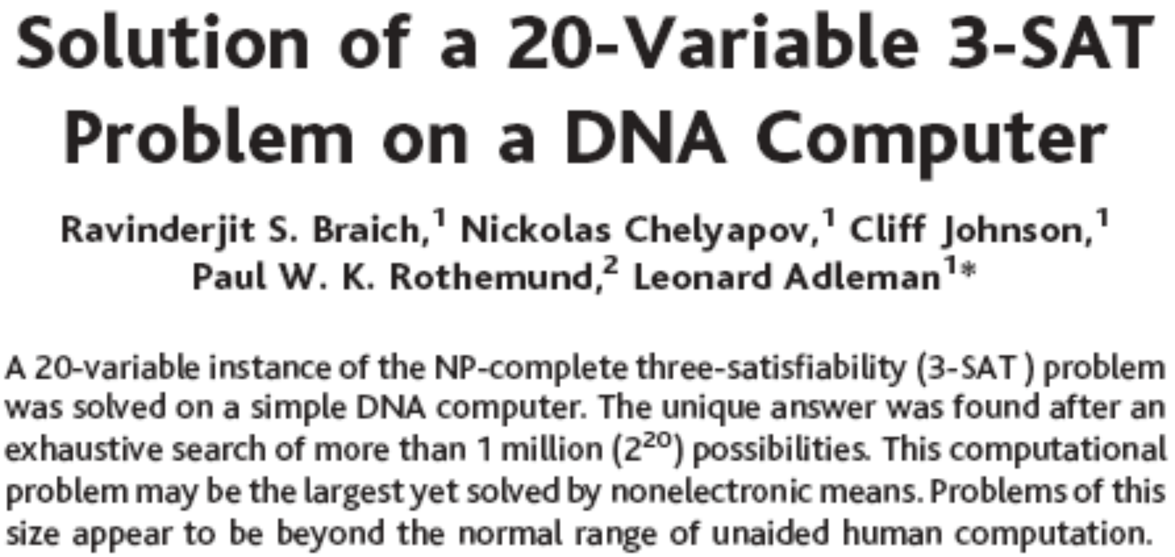


Fig. 3. Agarose gel electrophoresis of various products of the experiment. (**A**) Product of the ligation reaction (lane 1), PCR amplification of the product of the ligation reaction (lanes 2 through 5), and molecular weight marker in base pairs (lane 6). (**B**) Graduated PCR of the product from Step 3 (lanes 1 through 6); the molecular weight marker is in lane 7. (**C**) Graduated PCR of the final product of the experiment, revealing the Hamiltonian path (lanes 1 through 6); the molecular weight marker is in lane 7.

- Step 1: Form paths by annealing & ligating node strands and edge strands.
- Step 2: Amplify using v_{in} and v_{out} PCR primers.
- Step 3: Gel separate correct size (140 bp); amplify.
- Step 4: Discard paths missing any node (biotin-DNA, streptavidin-magnetic bead selections).
- Step 5: HP path exists if path molecules remain.

Adleman

- SPEED:
  - Desktop computer (1994) 10^6 operations/sec.
  - Supercomputer 10^{12} ops/sec.
  - Adleman 10^{14} ops/sec (Ligation = operation).
    - Could push to 10^{20} ops/sec.
- ENERGY CONSUMPTION:
  - Ligation: ATP ==> ADP + pP. DG = -8 kcal/mol
  - 1 joule provides $2x10^{19}$ ligations. $2x10^{19}$ ops/joule.
  - Theoretical maximum = $34x10^{19}$ ops/joule.
  - Electronic computer = 10^9 ops/joule.
- STORAGE SPACE:
  - DNA = 1 bit/nm^3
  - Video tape = 1 bit per 10^{12} nm^3
  - DNA is 12 orders of magnitude more compact.
- COST:
  - DNA molecule costs 100 femtocents...



Solution of a 20-Variable 3-SAT Problem on a DNA Computer

Ravinderjit S. Braich,[1] Nickolas Chelyapov,[1] Cliff Johnson,[1] Paul W. K. Rothemund,[2] Leonard Adleman[1]*

A 20-variable instance of the NP-complete three-satisfiability (3-SAT) problem was solved on a simple DNA computer. The unique answer was found after an exhaustive search of more than 1 million (2^{20}) possibilities. This computational problem may be the largest yet solved by nonelectronic means. Problems of this size appear to be beyond the normal range of unaided human computation.

SCIENCE VOL 296 19 APRIL 2002

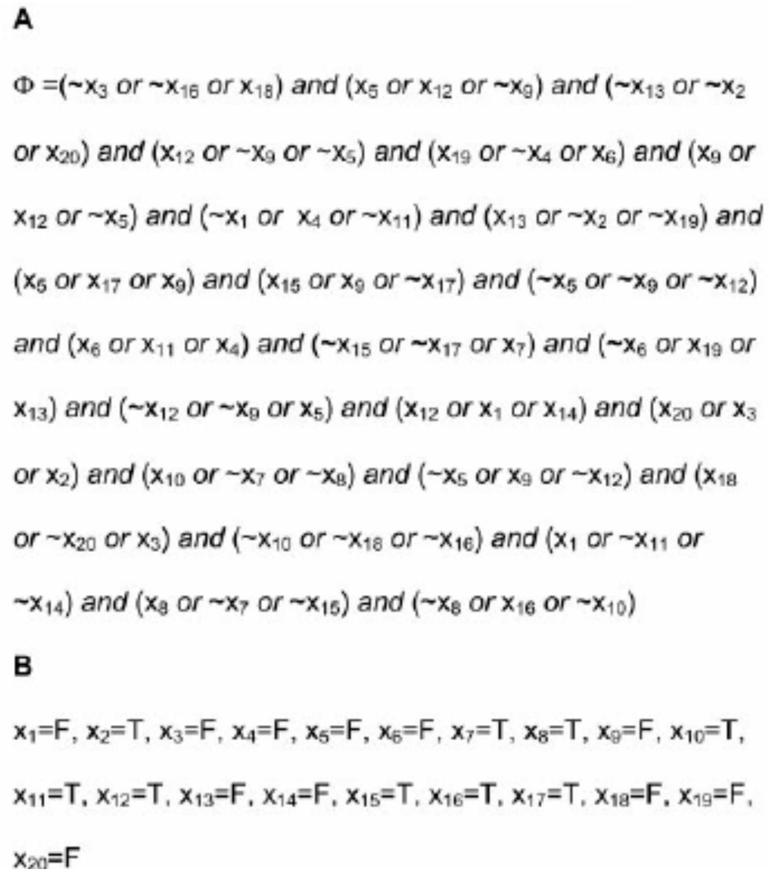
Satisfiability Problem (SAT):

Given a Boolean fomula find a truth assignment for all variables such that the entire formula is satisfied

(Answer = 1).

3-SAT. 3-CNF

(conjunctive normal form).

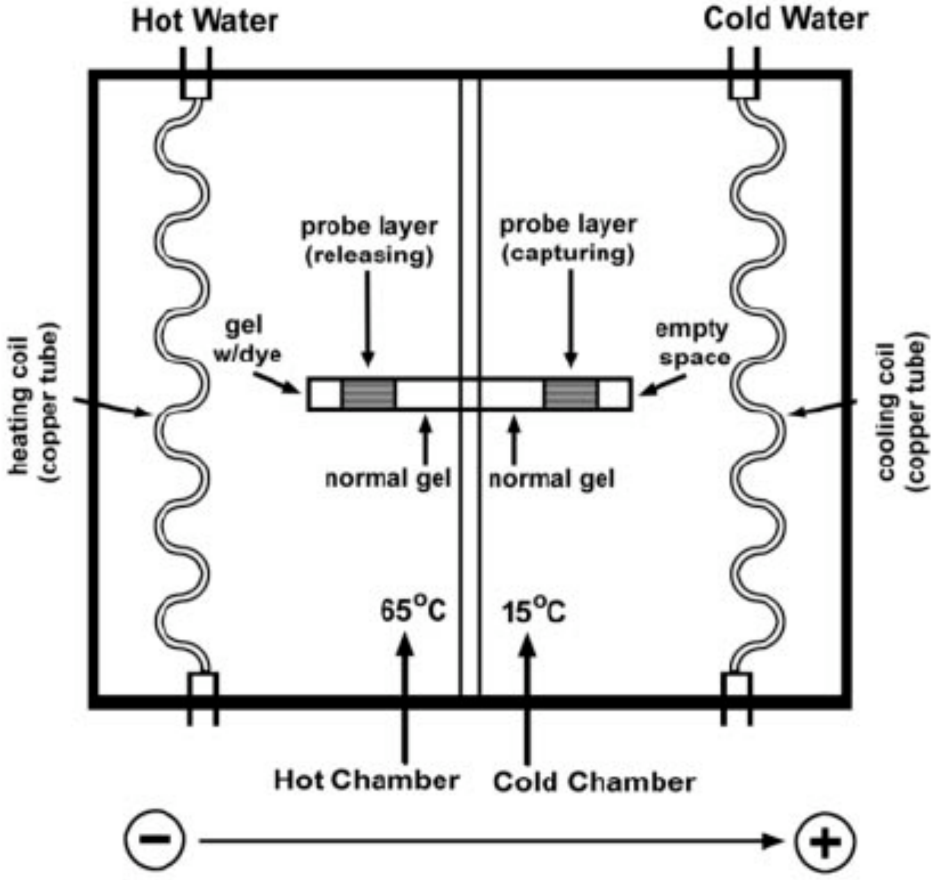


A

$$\Phi =(\sim x_3 \textit{ or } \sim x_{16} \textit{ or } x_{18}) \textit{ and } (x_5 \textit{ or } x_{12} \textit{ or } \sim x_9) \textit{ and } (\sim x_{13} \textit{ or } \sim x_2 \textit{ or } x_{20}) \textit{ and } (x_{12} \textit{ or } \sim x_9 \textit{ or } \sim x_5) \textit{ and } (x_{19} \textit{ or } \sim x_4 \textit{ or } x_6) \textit{ and } (x_9 \textit{ or } x_{12} \textit{ or } \sim x_5) \textit{ and } (\sim x_1 \textit{ or } x_4 \textit{ or } \sim x_{11}) \textit{ and } (x_{13} \textit{ or } \sim x_2 \textit{ or } \sim x_{19}) \textit{ and } (x_5 \textit{ or } x_{17} \textit{ or } x_9) \textit{ and } (x_{15} \textit{ or } x_9 \textit{ or } \sim x_{17}) \textit{ and } (\sim x_5 \textit{ or } \sim x_9 \textit{ or } \sim x_{12}) \textit{ and } (x_6 \textit{ or } x_{11} \textit{ or } x_4) \textit{ and } (\sim x_{15} \textit{ or } \sim x_{17} \textit{ or } x_7) \textit{ and } (\sim x_6 \textit{ or } x_{19} \textit{ or } x_{13}) \textit{ and } (\sim x_{12} \textit{ or } \sim x_9 \textit{ or } x_5) \textit{ and } (x_{12} \textit{ or } x_1 \textit{ or } x_{14}) \textit{ and } (x_{20} \textit{ or } x_3 \textit{ or } x_2) \textit{ and } (x_{10} \textit{ or } \sim x_7 \textit{ or } \sim x_8) \textit{ and } (\sim x_5 \textit{ or } x_9 \textit{ or } \sim x_{12}) \textit{ and } (x_{18} \textit{ or } \sim x_{20} \textit{ or } x_3) \textit{ and } (\sim x_{10} \textit{ or } \sim x_{18} \textit{ or } \sim x_{16}) \textit{ and } (x_1 \textit{ or } \sim x_{11} \textit{ or } \sim x_{14}) \textit{ and } (x_8 \textit{ or } \sim x_7 \textit{ or } \sim x_{15}) \textit{ and } (\sim x_8 \textit{ or } x_{16} \textit{ or } \sim x_{10})$$

B

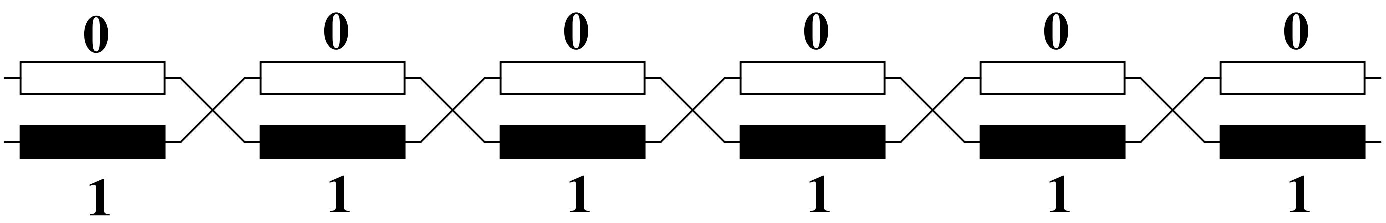
x_1=F, x_2=T, x_3=F, x_4=F, x_5=F, x_6=F, x_7=T, x_8=T, x_9=F, x_{10}=T, x_{11}=T, x_{12}=T, x_{13}=F, x_{14}=F, x_{15}=T, x_{16}=T, x_{17}=T, x_{18}=F, x_{19}=F, x_{20}=F

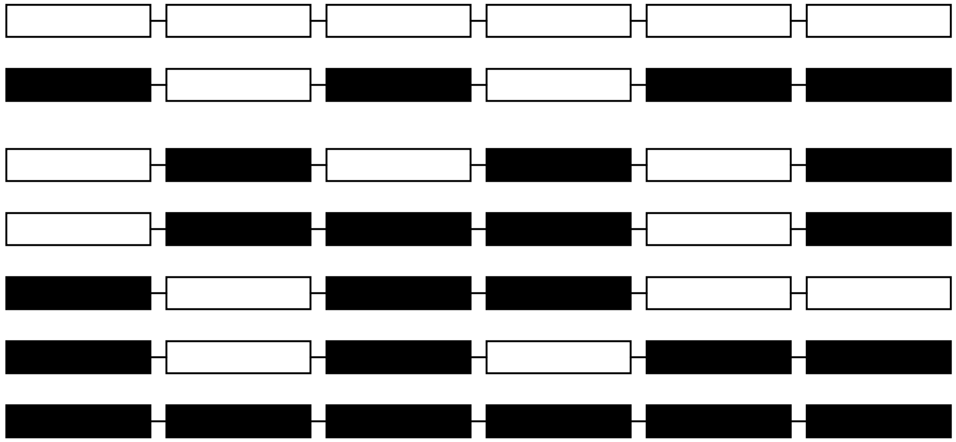


- Required new experimental setup.
  - Acrydite-modified oligonucleotides, immobilization of library strands by acrylamide bound probe strands.
- 1,048,576 possible truth assignments.
- Half-length library strands synthesized by mix-and-split solid phase phosphoramidite methods.

SCIENCE VOL 296 19 APRIL 2002 499

Mix-and-split Synthesis



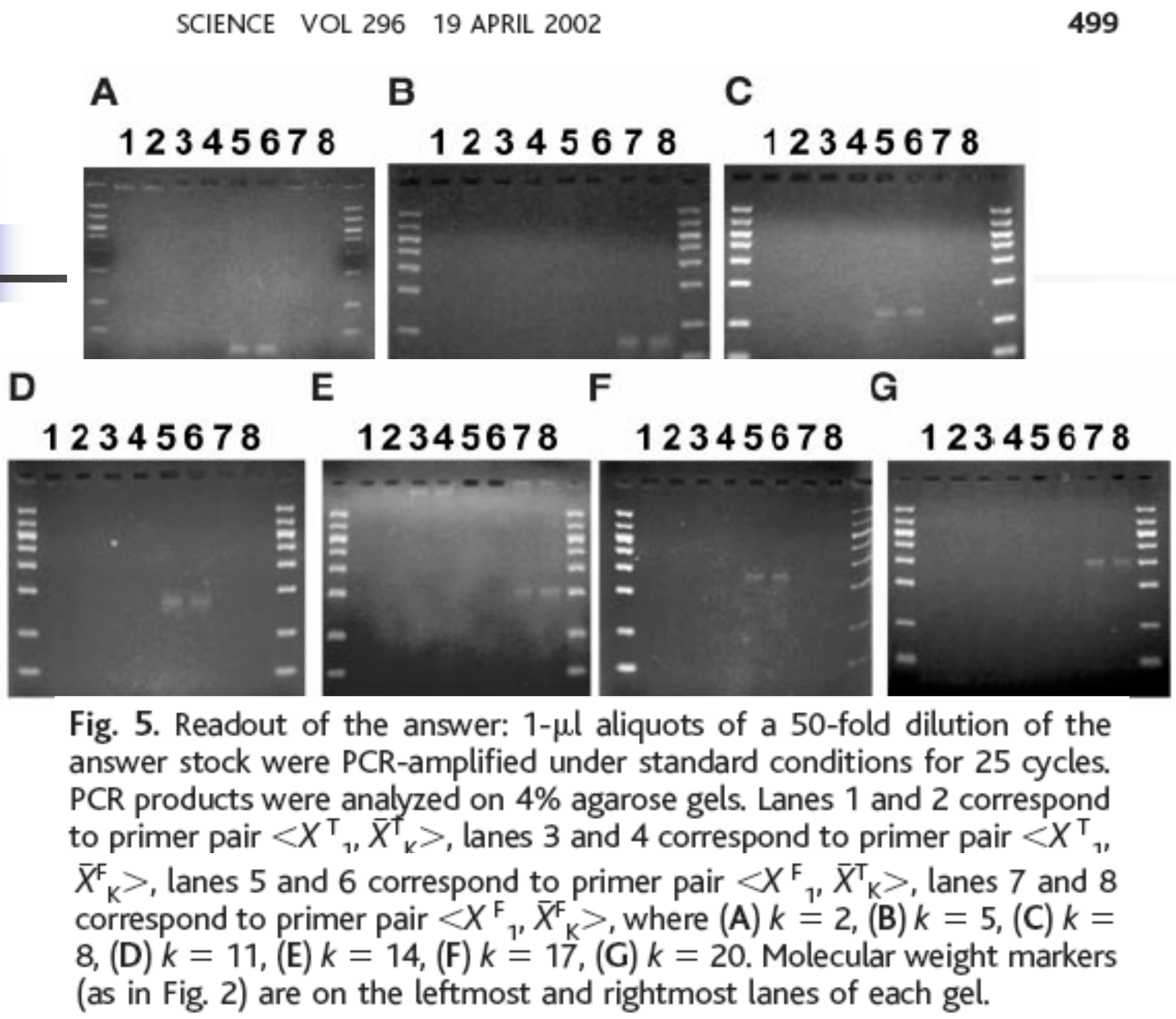


Resulting sequence library.

SCIENCE VOL 296 19 APRIL 2002 499

Adleman 20 variable SAT

- STEP 1: Synthesize/ligate assignment strands.
- STEP 2: Electrophoresis in hot module to unbind and cool module to bind correct assignments to immobilized probes.
- STEP 3: Switch cool module to hot chamber and repeat unbind/bind step for next clause. (24 cycles)
- STEP 4: Extract answer; amplify; read out by PCR.



SCIENCE VOL 296 19 APRIL 2002 499

Fig. 5. Readout of the answer: 1-µl aliquots of a 50-fold dilution of the answer stock were PCR-amplified under standard conditions for 25 cycles. PCR products were analyzed on 4% agarose gels. Lanes 1 and 2 correspond to primer pair $<X^{T}_{1}, \bar{X}^{T}_{K}>$, lanes 3 and 4 correspond to primer pair $<X^{T}_{1}, \bar{X}^{F}_{K}>$, lanes 5 and 6 correspond to primer pair $<X^{F}_{1}, \bar{X}^{T}_{K}>$, lanes 7 and 8 correspond to primer pair $<X^{F}_{1}, \bar{X}^{F}_{K}>$, where (**A**) $k = 2$, (**B**) $k = 5$, (**C**) $k = 8$, (**D**) $k = 11$, (**E**) $k = 14$, (**F**) $k = 17$, (**G**) $k = 20$. Molecular weight markers (as in Fig. 2) are on the leftmost and rightmost lanes of each gel.

S. Roweis et al., *J. Comp. Biol.* **5**, 615 (1998).

Sticker Model

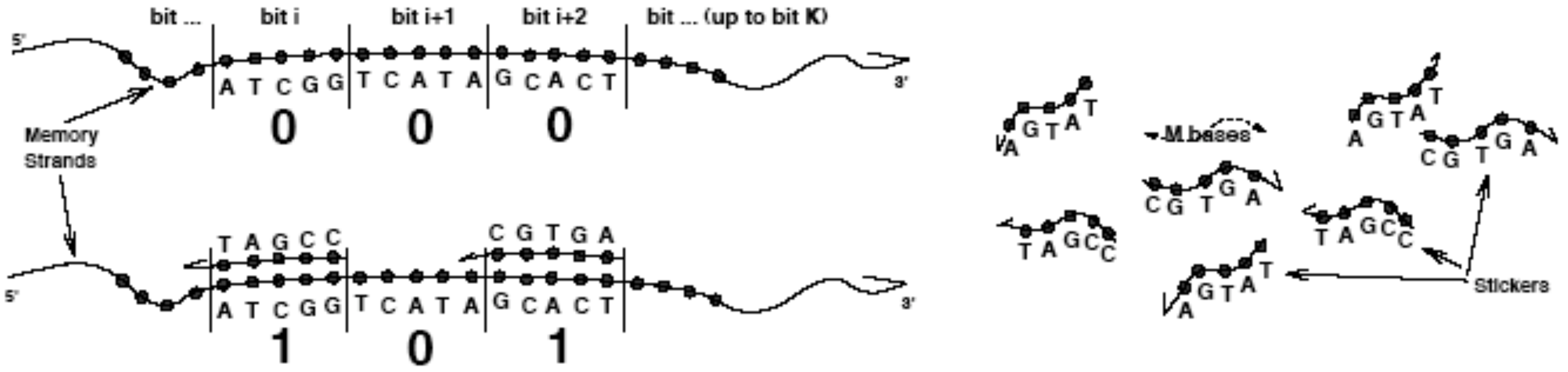
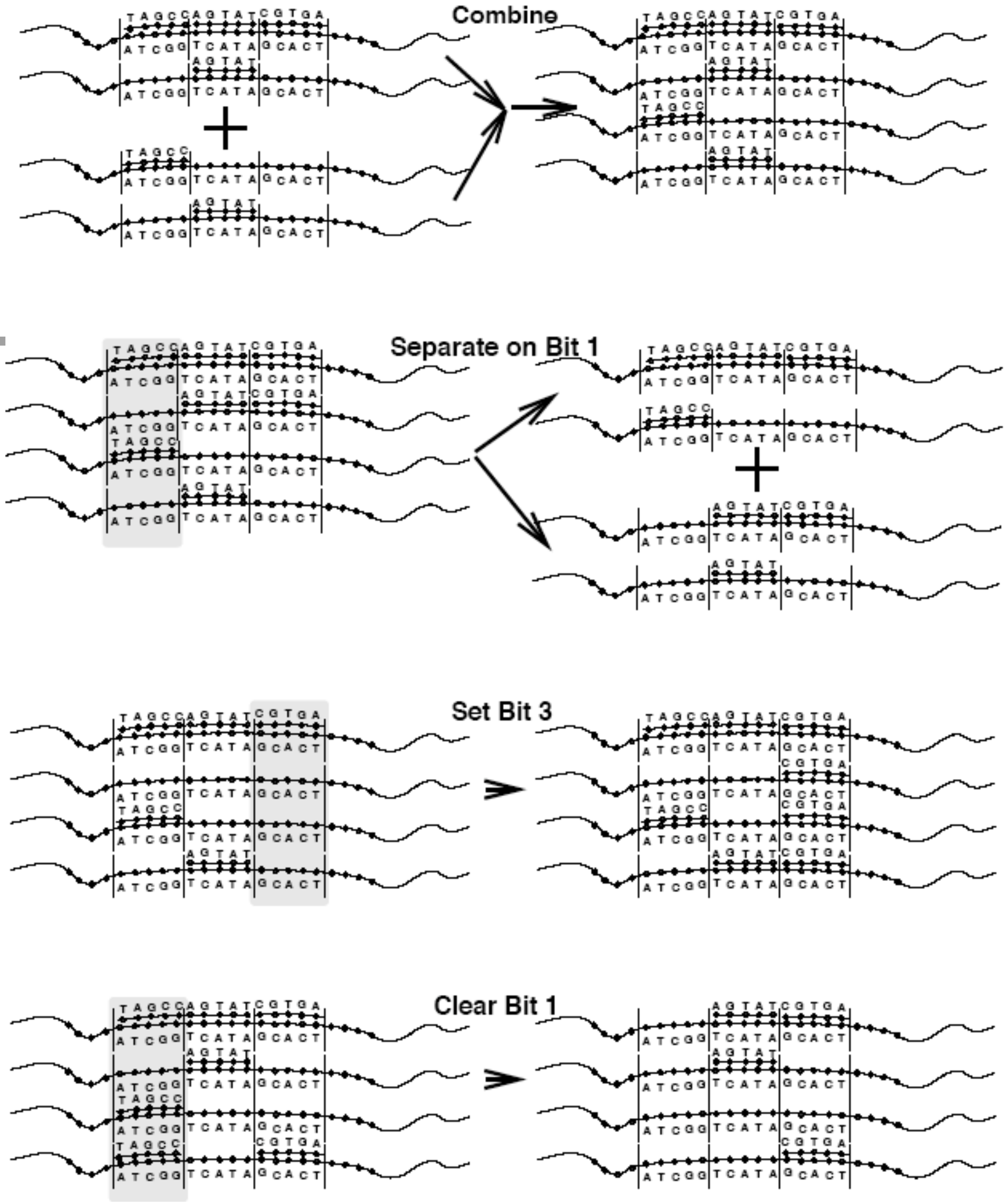


Figure 1: A memory strand and associated stickers (together called a *memory complex*) represent a bit string. The top complex on the left has all three bits *off*; the bottom complex has two annealed stickers and thus two bits *on*.

- All sequences always present.
- Values in bit string represent by presence or absence of an annealed sticker oligo.

S. Roweis et al., *J. Comp. Biol.* **5**, 615 (1998).

Sticker Model



- 4 basic operations.
- Combine by pipetting.
- Separate by annealing to probe.
- Set by adding stickers.
- Clear by adding PNA.

S. Roweis et al., *J. Comp. Biol.* **5**, 615 (1998).

Sticker Model

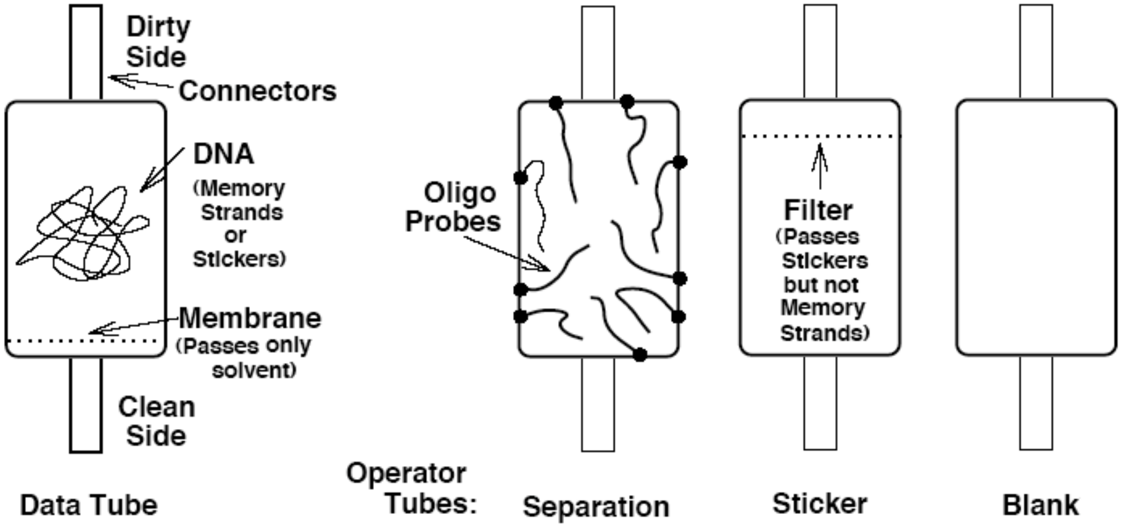


Figure 3: Data and Operator Tubes in the stickers machine.

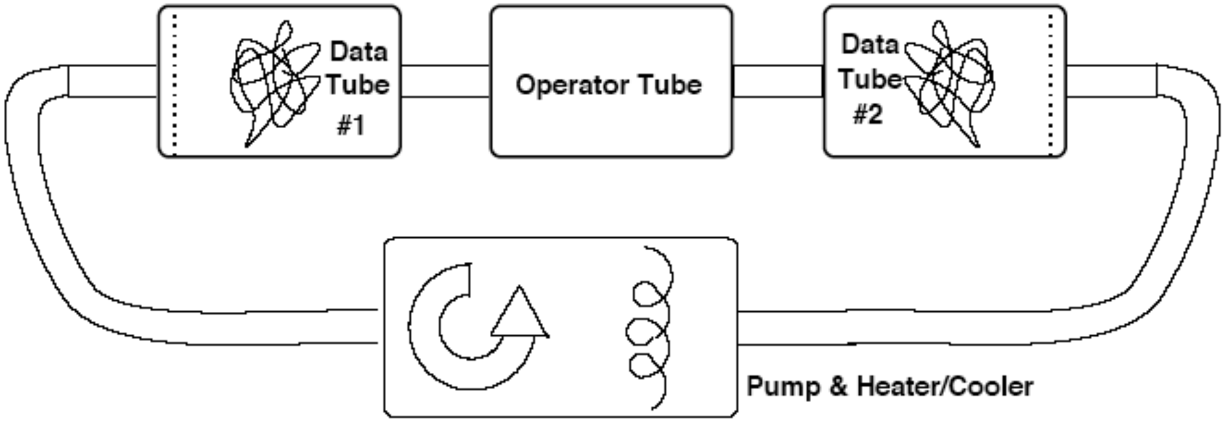
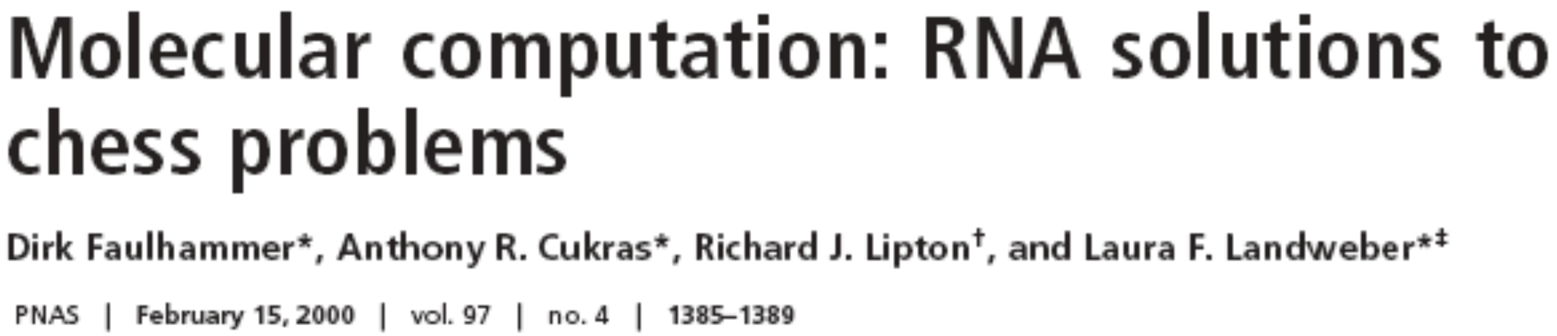


Figure 4: Setup for a generic operation in the stickers machine.



Molecular computation: RNA solutions to chess problems

Dirk Faulhammer*, Anthony R. Cukras*, Richard J. Lipton†, and Laura F. Landweber*‡

PNAS | February 15, 2000 | vol. 97 | no. 4 | 1385–1389

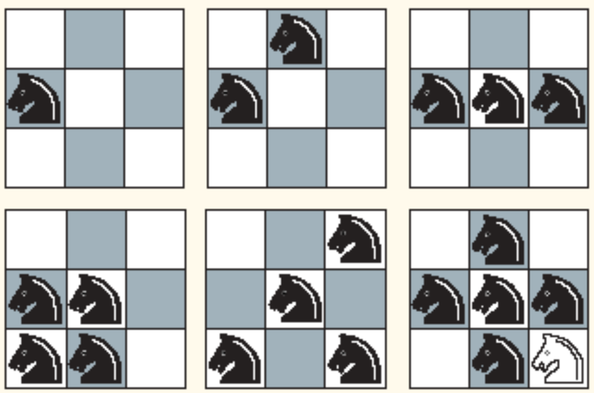
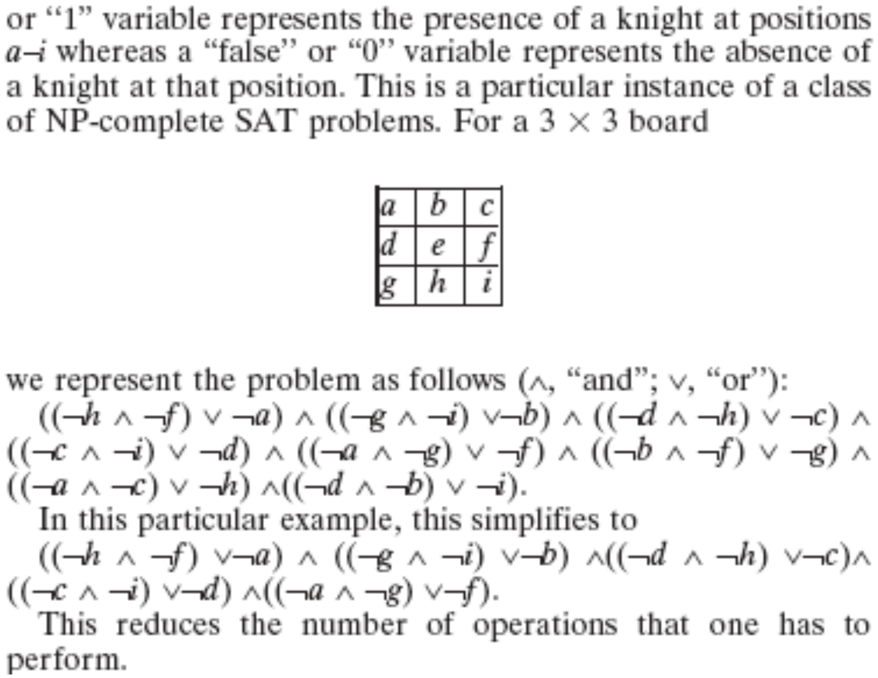


Figure 4 | **Five correct solutions to the Knight Problem and one incorrect solution.** In the last solution, the white knight in square i (see FIG. 3 for key) threatens the knights in squares b and d.



or "1" variable represents the presence of a knight at positions *a*–*i* whereas a "false" or "0" variable represents the absence of a knight at that position. This is a particular instance of a class of NP-complete SAT problems. For a 3 × 3 board

a	*b*	*c*
d	*e*	*f*
g	*h*	*i*

we represent the problem as follows (∧, "and"; ∨, "or"):

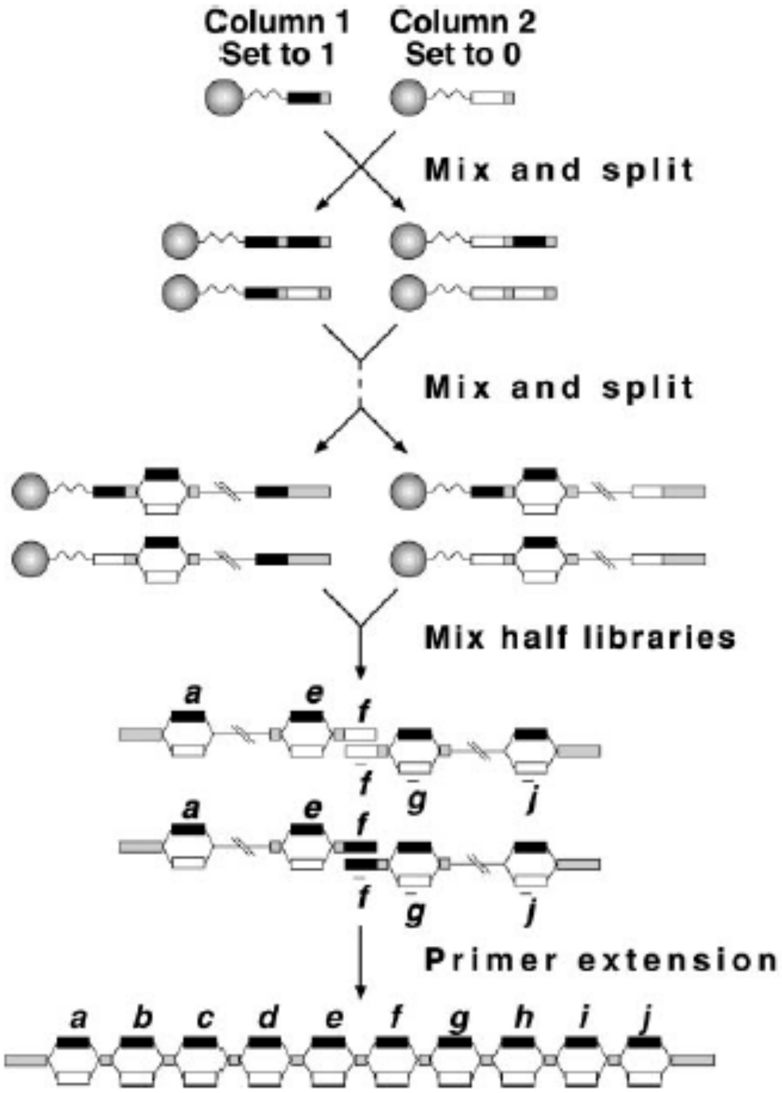
$((\neg h \wedge \neg f) \vee \neg a) \wedge ((\neg g \wedge \neg i) \vee \neg b) \wedge ((\neg d \wedge \neg h) \vee \neg c) \wedge ((\neg c \wedge \neg i) \vee \neg d) \wedge ((\neg a \wedge \neg g) \vee \neg f) \wedge ((\neg b \wedge \neg f) \vee \neg g) \wedge ((\neg a \wedge \neg c) \vee \neg h) \wedge ((\neg d \wedge \neg b) \vee \neg i)$.

In this particular example, this simplifies to

$((\neg h \wedge \neg f) \vee \neg a) \wedge ((\neg g \wedge \neg i) \vee \neg b) \wedge ((\neg d \wedge \neg h) \vee \neg c) \wedge ((\neg c \wedge \neg i) \vee \neg d) \wedge ((\neg a \wedge \neg g) \vee \neg f)$.

This reduces the number of operations that one has to perform.

- RNA molecular encodings
- Mark sequence with DNA primer and digest with Rnase H
- Readout by multiplex linear PCR

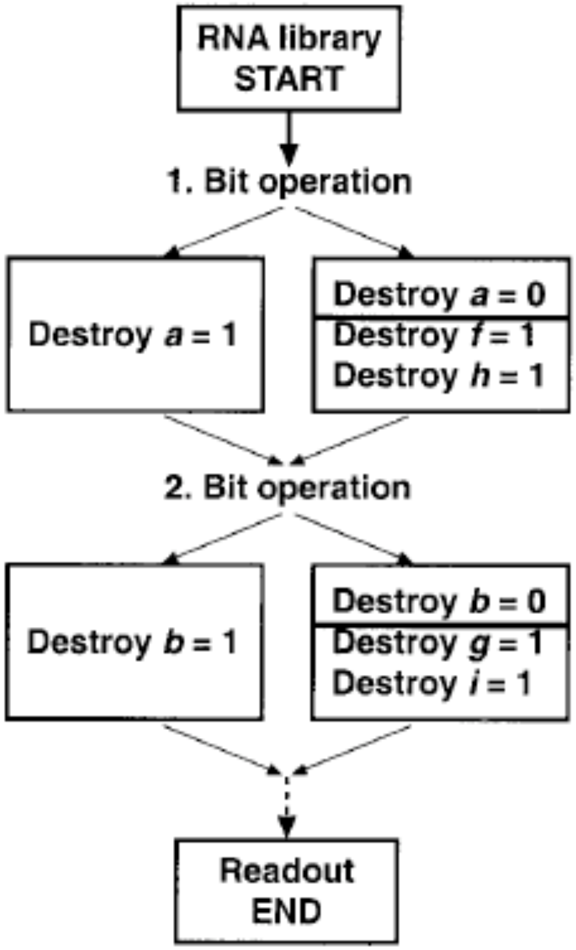


PNAS | February 15, 2000 | vol. 97 | no. 4 | 1385–1389

- DNA synthesis.
- 10 bits available.
- 25 cycles of PCR resulted in “bit shuffled” answer strands. 15 cycles no bit shuffling seen. 5 cycles used in experiments.

5′ Prefix	Bit *a* (0/1)	Spacer 1	Bit *b* (0/1)	Spacer 2	. . .	Spacer 9	Bit *j* (0/1)	Suffix 3′
24 nt	15 nt	5 nt	15 nt	5 nt	. . .	5 nt	15 nt	32 nt

PNAS | February 15, 2000 | vol. 97 | no. 4 | 1385–1389



RNA library START

1. Bit operation

Destroy *a* = 1

Destroy *a* = 0
Destroy *f* = 1
Destroy *h* = 1

2. Bit operation

Destroy *b* = 1

Destroy *b* = 0
Destroy *g* = 1
Destroy *i* = 1

Readout END

a	*b*	*c*
d	*e*	*f*
g	*h*	*i*

$$((\neg h \wedge \neg f) \vee \neg a)$$

$$((\neg g \wedge \neg i) \vee \neg b)$$

Bit	Knight absent code, bit set to 0	Knight present code, bit set to 1	Spacer code
a	CTCTTACTCAATTCT	TCCTCACATTACTTA	TCTAC
b	CATATCAACATCTTA	ACTTCCTTTATATCC	ATAAC
c	ATCCTCCACTTCACA	TTATAACAAACATCC	CTTAA
d	TTAAAATCTTCCCTC	ACATAACCCTCTTCA	TTTAC
e	CTATTTATCCACACC	ACCTTACTTTCCATA	TACAA
f	GCTTCAAACAATTCC	GTACATTCTCCCTAC	TCCTT
g	AACTCTCAAATTCAA	CATAATCTTATATTC	TCAAT
h	CTAACCTTTACTTCA	ATAATCACATACTTC	TCCAA
i	CATTCCTTATCCCAC	TCCACCAACTACCTA	ACACA
j	CACCCTTTCTCCTCT	TTTTAAATTTCACAA	SUFFIX

PNAS | February 15, 2000 | vol. 97 | no. 4 | 1385–1389

RNA Computer

- RNA molecular encodings
- Mark sequence with DNA primer and digest with Rnase H
- Readout by multiplex linear PCR

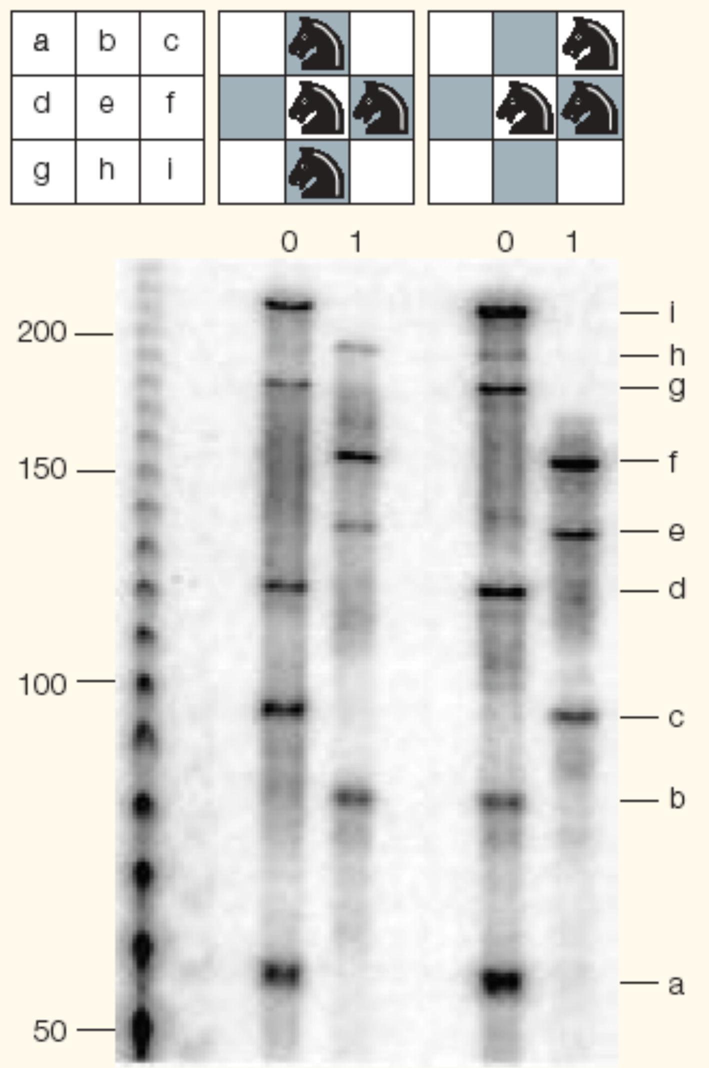
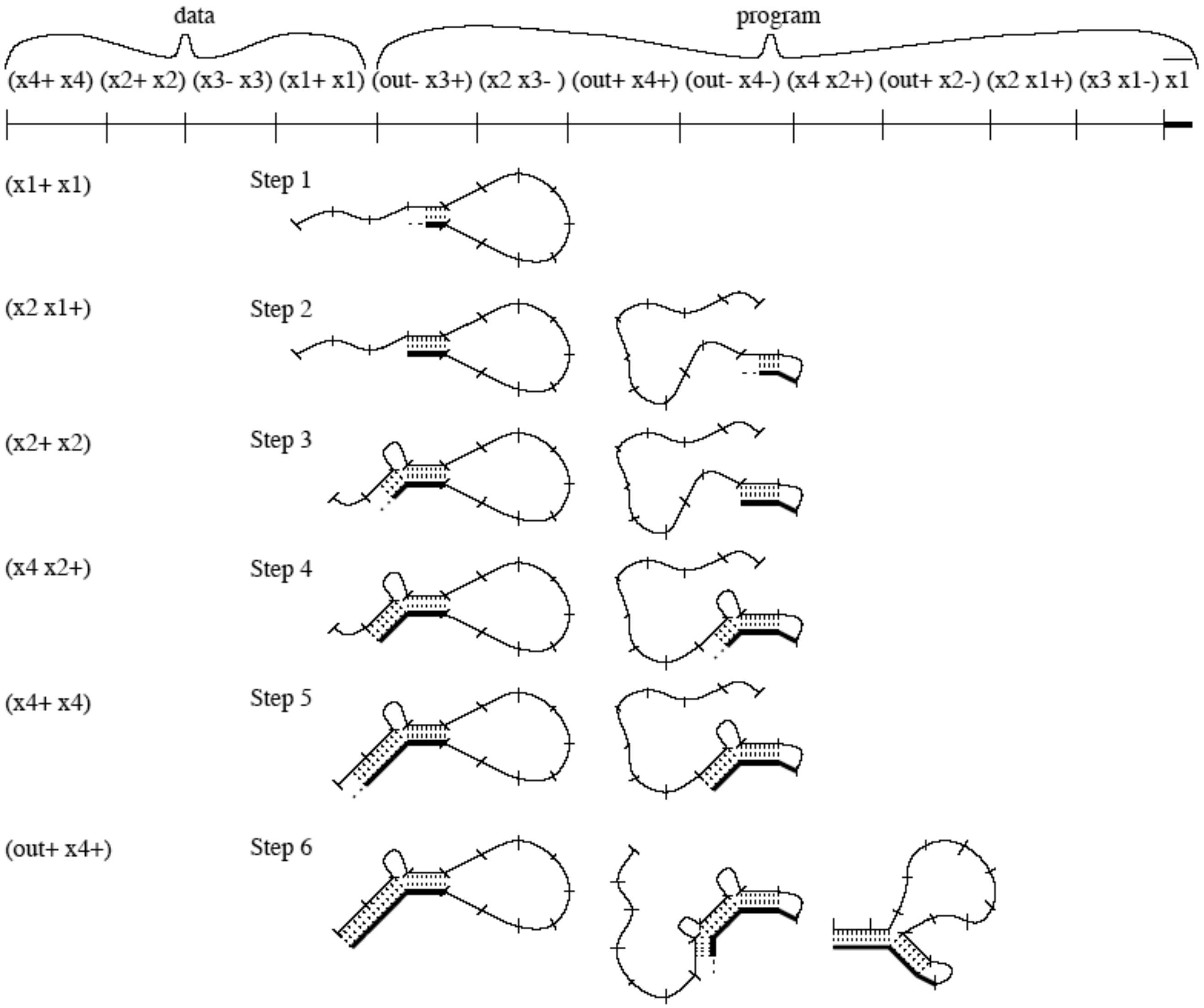
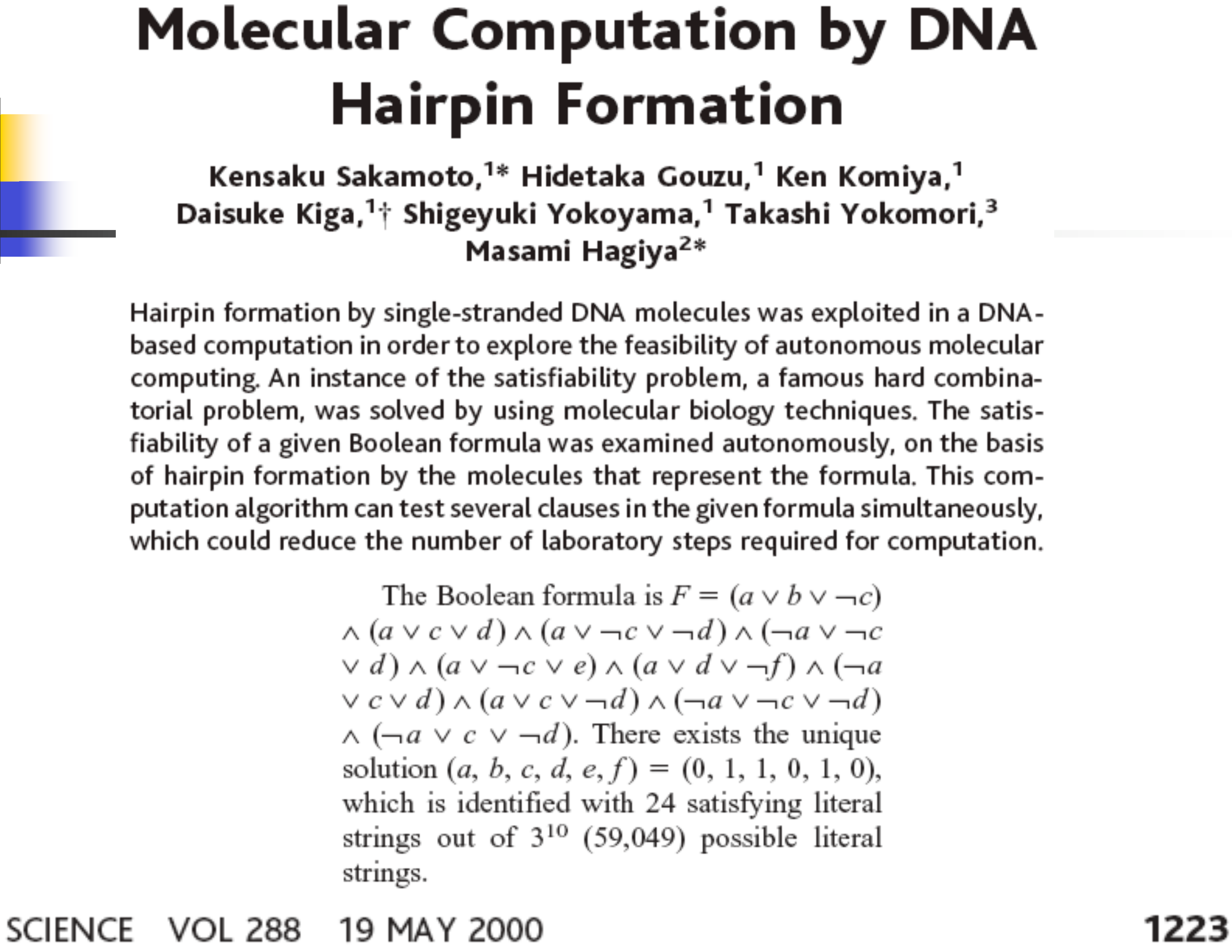


Figure 3 | **Solving the Knight Problem.** Multiplex linear PCR produces a 'bar-code' representing two configurations of knights on a 3 × 3 chessboard[3].

Whiplash PCR for $O(1)$ Computing

Erik Winfree


data
program
(x4+ x4) (x2+ x2) (x3- x3) (x1+ x1) (out- x3+) (x2 x3-) (out+ x4+) (out- x4-) (x4 x2+) (out+ x2-) (x2 x1+) (x3 x1-) x1
(x1+ x1)
Step 1
(x2 x1+)
Step 2
(x2+ x2)
Step 3
(x4 x2+)
Step 4
(x4+ x4)
Step 5
(out+ x4+)
Step 6



Molecular Computation by DNA Hairpin Formation

Kensaku Sakamoto,[1*] Hidetaka Gouzu,[1] Ken Komiya,[1] Daisuke Kiga,[1†] Shigeyuki Yokoyama,[1] Takashi Yokomori,[3] Masami Hagiya[2*]

Hairpin formation by single-stranded DNA molecules was exploited in a DNA-based computation in order to explore the feasibility of autonomous molecular computing. An instance of the satisfiability problem, a famous hard combinatorial problem, was solved by using molecular biology techniques. The satisfiability of a given Boolean formula was examined autonomously, on the basis of hairpin formation by the molecules that represent the formula. This computation algorithm can test several clauses in the given formula simultaneously, which could reduce the number of laboratory steps required for computation.

The Boolean formula is $F = (a \vee b \vee \neg c) \wedge (a \vee c \vee d) \wedge (a \vee \neg c \vee \neg d) \wedge (\neg a \vee \neg c \vee d) \wedge (a \vee \neg c \vee e) \wedge (a \vee d \vee \neg f) \wedge (\neg a \vee c \vee d) \wedge (a \vee c \vee \neg d) \wedge (\neg a \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee \neg d)$. There exists the unique solution $(a, b, c, d, e, f) = (0, 1, 1, 0, 1, 0)$, which is identified with 24 satisfying literal strings out of 3^{10} (59,049) possible literal strings.

Hairpin formation

“Literal strings” were introduced to encode the given formula; they are conjunctions of the literals selected from each clause (one literal per clause). For example, the formula $(a \vee b) \wedge (\neg a \vee \neg c)$ can be represented by the set of four literal strings, a- $\neg a$, a- $\neg c$, b- $\neg a$, and b- $\neg c$. A formula is satisfiable if there is such a literal string that does not involve any variable together with its negation (we refer to such strings as “satisfying”), because the literal string gives a value assignment that makes every clause take the value 1 simultaneously, so that the value of the entire formula will become 1. If each variable is encoded with a sequence complementary to that encoding its negation, then the literal strings containing one or more pairs of complementary literals can form hairpins, and the satisfying strings stay in the form without such hairpins (“nonhairpin” form) (Fig. 1).

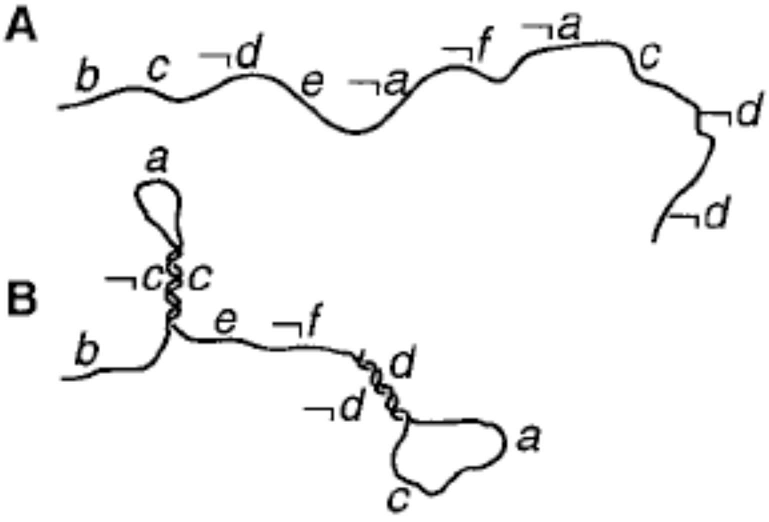


Fig. 1. Illustration of the computation based on DNA hairpin formation. **(A)** The ssDNA representing a satisfying literal string, *b-c-* ¬*d-e-* ¬*a-* ¬*f-* ¬*a-c-* ¬*d-* ¬*d*, stays in the nonhairpin form. **(B)** A literal string, *b-* ¬*c-a-c-e-* ¬*f-* ¬*d-c-a-d*, has pairs of the complementary literals (*c-* ¬*c* and ¬*d-d*), and the ssDNA representing it forms hairpins.

SCIENCE VOL 288 19 MAY 2000 1223

Hairpin formation

Generation of “literal” strands.

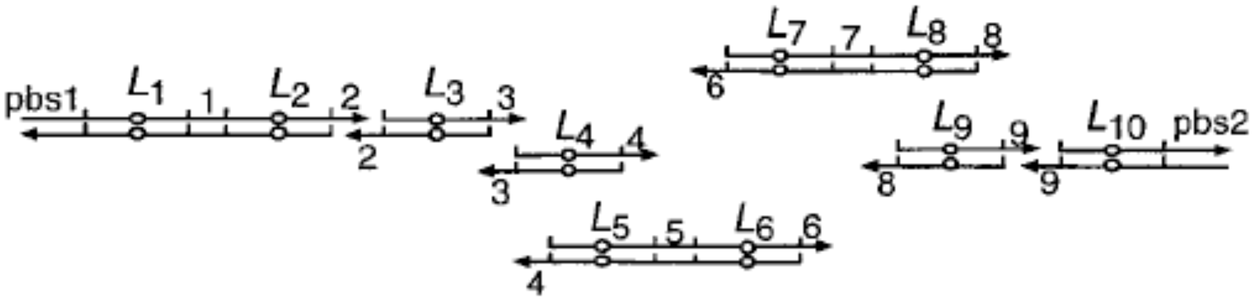


Fig. 2. Illustration of “literal units” in the course of the assembly into a 10-clause or full-length literal string. The arrows represent ssDNA molecules; their direction is from the 5′ to 3′ ends. L_k indicates the literal from clause k ($1 \leq k \leq 10$), and the numbers at both ends of each literal indicate the linker numbers. The circles in the middle of the literals represent the Bst NI sites, and pbs1 and pbs2 indicate the primer binding sites for PCR.

Hairpin removal step substitutes:

Destroy hairpin strands with restriction enzyme.

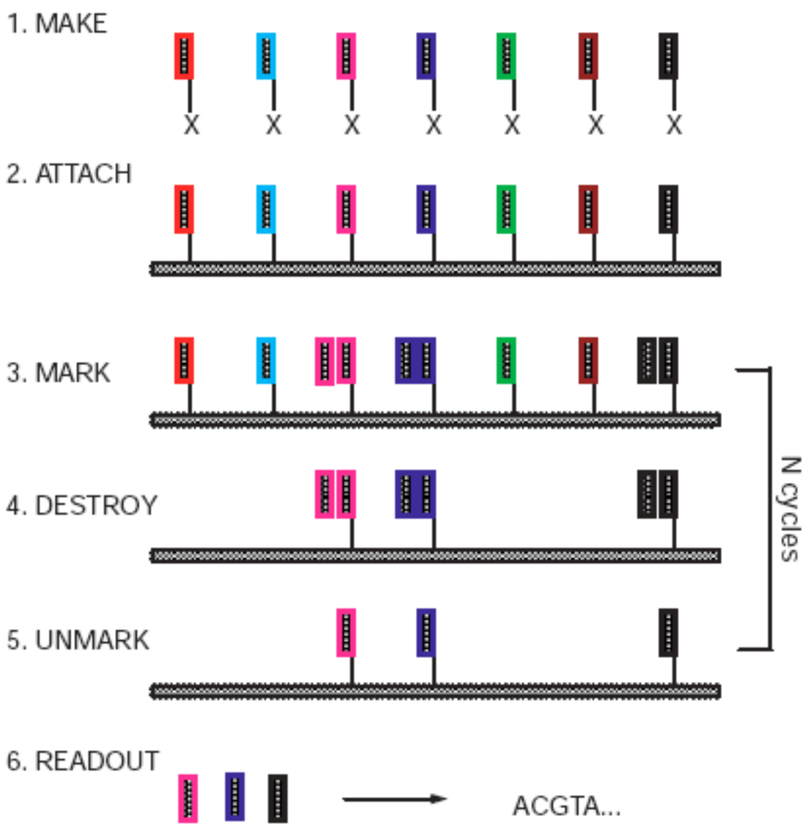
Amplify non-hairpin strands to dilute hairpins.

DNA computing on surfaces

Qinghua Liu*†, Liman Wang*, Anthony G. Frutos*†, Anne E. Condon†‡,
Robert M. Corn* & Lloyd M. Smith*

* *Department of Chemistry,* ‡ *Department of Computer Science, University of Wisconsin, Madison, Wisconsin 53706, USA*

- SAT problem.
- 16 NT words 5'-FFFFvvvvvvvvFFFF-3'
- Operations: Mark, destroy, unmark.
- E.coli exonuclease I digests ssDNA (unmarked seqs).



$$(w \vee x \vee y) \wedge (w \vee \bar{y} \vee z) \wedge (\bar{x} \vee y) \wedge (\bar{w} \vee \bar{y})$$

NATURE | VOL 403 | 13 JANUARY 2000 |

Surface Computing

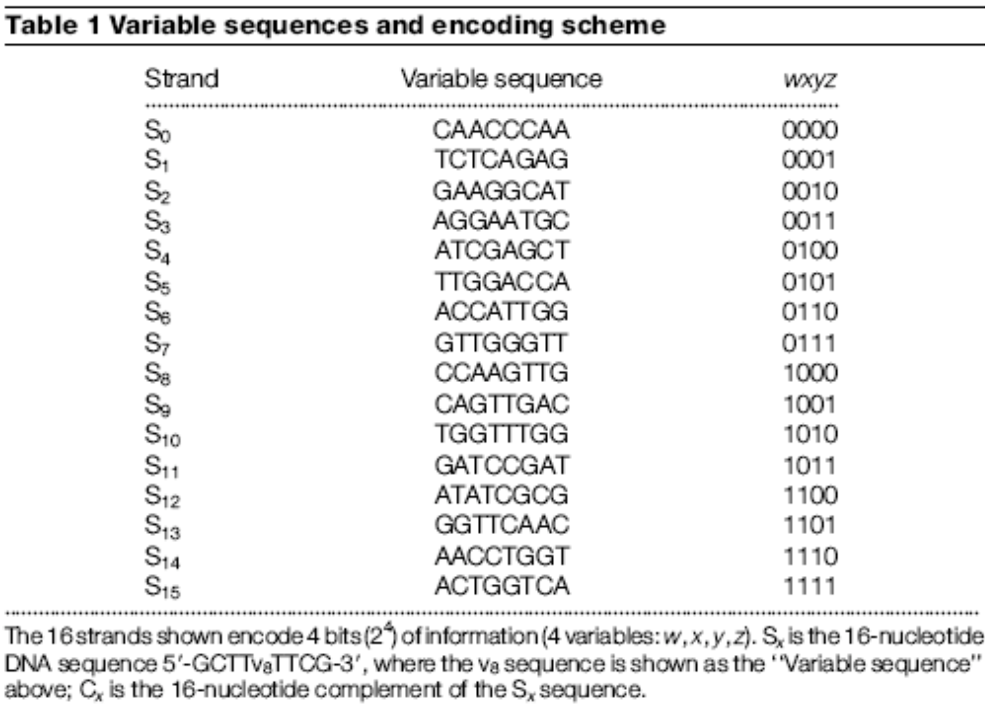
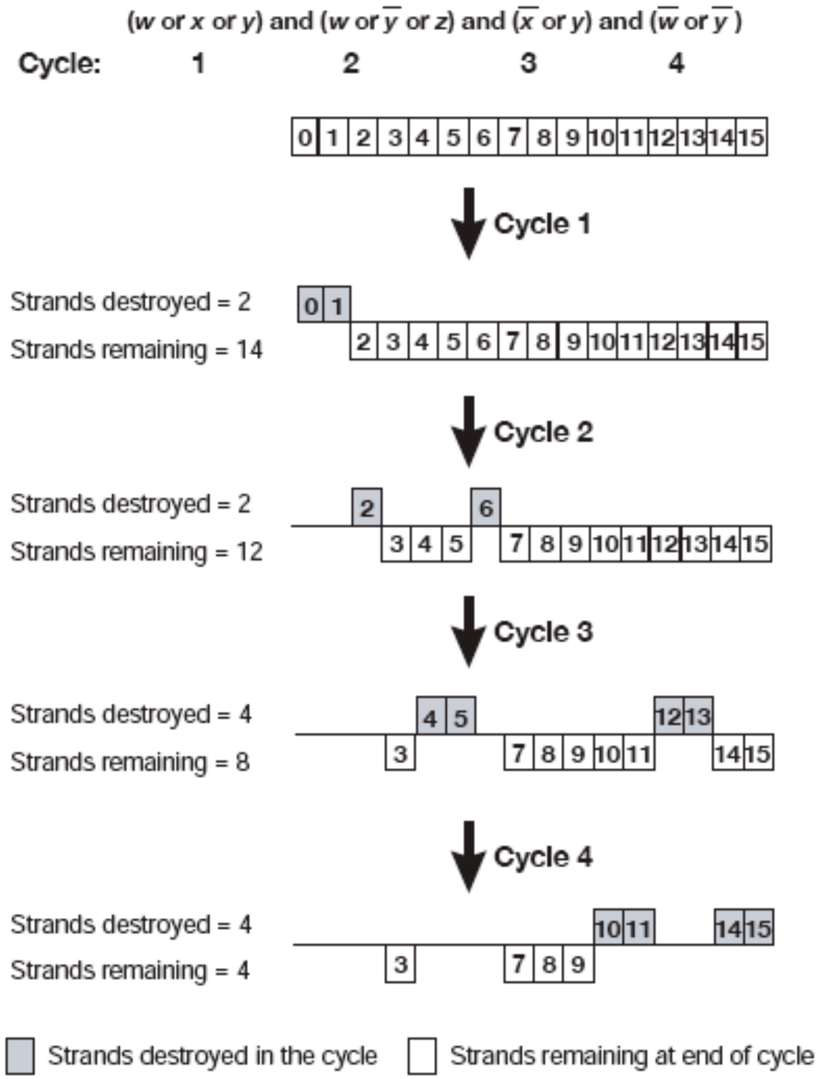


Table 1 Variable sequences and encoding scheme

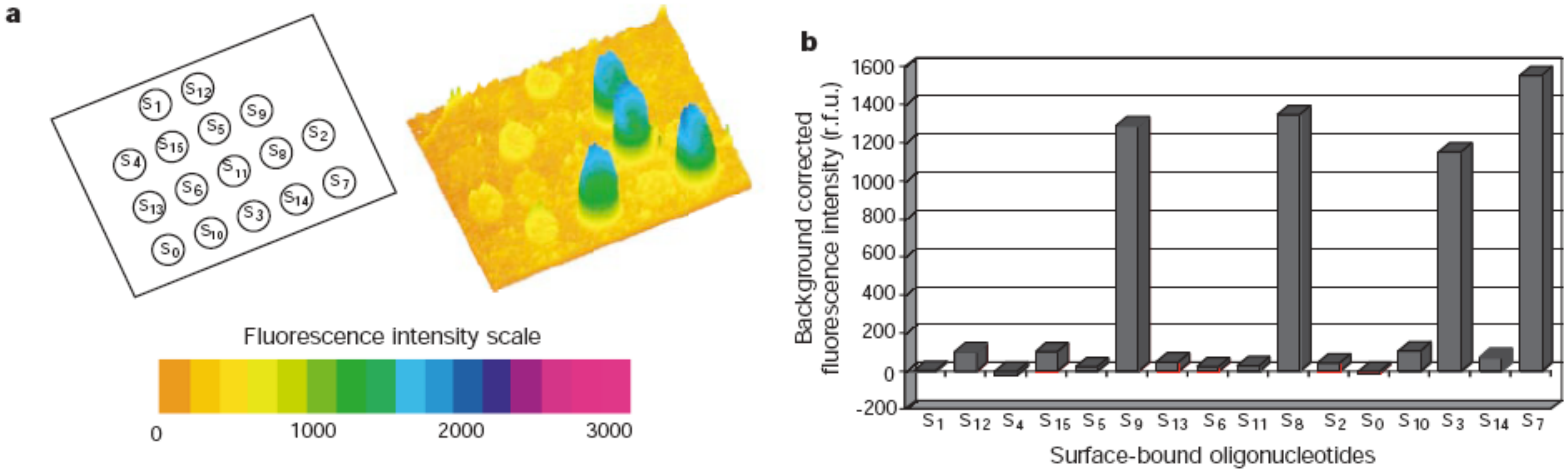
Strand	Variable sequence	*wxyz*
S_0	CAACCCAA	0000
S_1	TCTCAGAG	0001
S_2	GAAGGCAT	0010
S_3	AGGAATGC	0011
S_4	ATCGAGCT	0100
S_5	TTGGACCA	0101
S_6	ACCATTGG	0110
S_7	GTTGGGTT	0111
S_8	CCAAGTTG	1000
S_9	CAGTTGAC	1001
S_{10}	TGGTTTGG	1010
S_{11}	GATCCGAT	1011
S_{12}	ATATCGCG	1100
S_{13}	GGTTCAAC	1101
S_{14}	AACCTGGT	1110
S_{15}	ACTGGTCA	1111

The 16 strands shown encode 4 bits (2^4) of information (4 variables: w, x, y, z). S_x is the 16-nucleotide DNA sequence 5′-GCTTv$_8$TTCG-3′, where the v$_8$ sequence is shown as the ''Variable sequence'' above; C_x is the 16-nucleotide complement of the S_x sequence.



NATURE | VOL 403 | 13 JANUARY 2000

Surface Computing



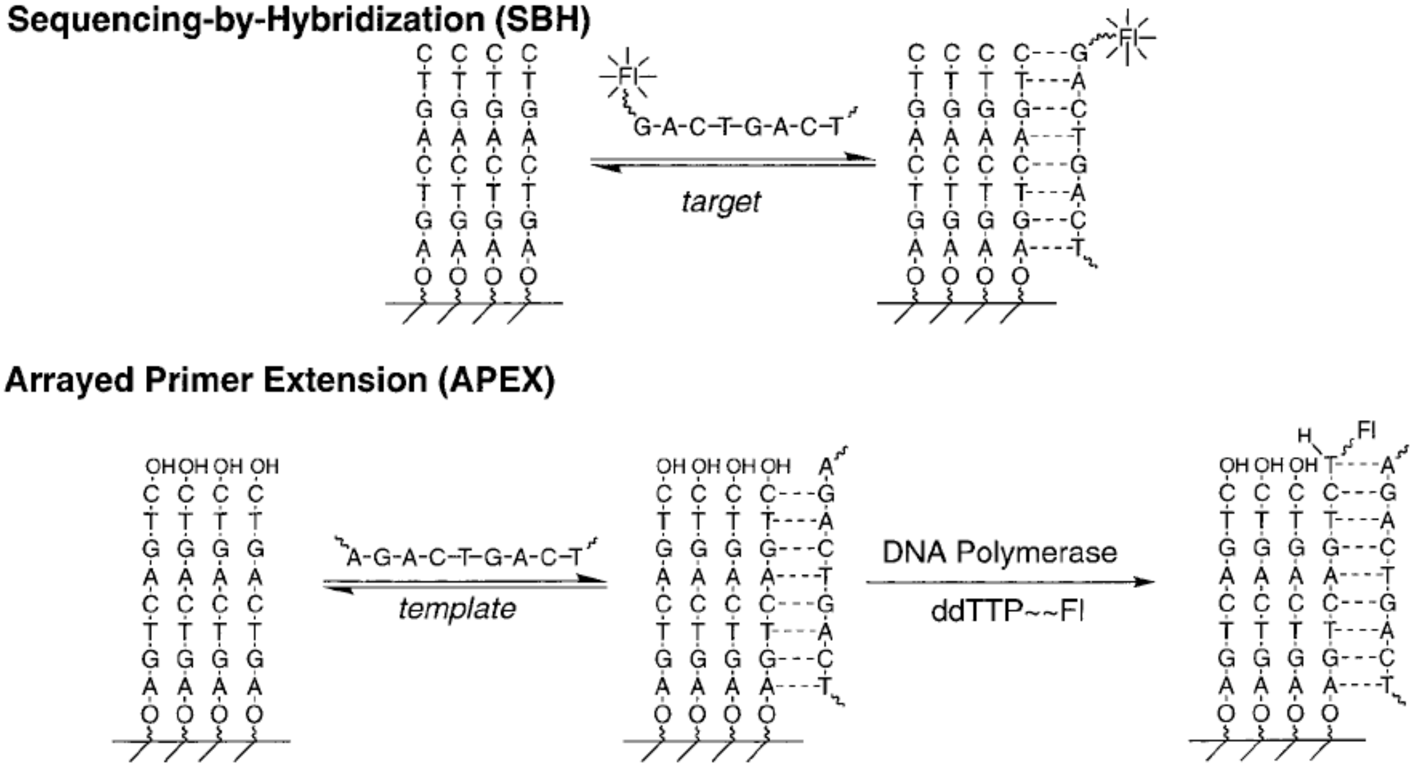
5′-biotin-ctgaatcttgtagatagcta-3′ 5′-FAM-tattttgagcagtggctcc-3′

Readout: PCR amplify results using primers above; Separate/remove biotinylated strands; anneal fluorescent strands back onto chip; read fluorescence.

NATURE | VOL 403 | 13 JANUARY 2000 |

The Arrayed Primer Extension Method for DNA Microchip Analysis. Molecular Computation of Satisfaction Problems

Michael C. Pirrung,*,† Richard V. Connors,† Amy L. Odenbaugh,† Michael P. Montague-Smith,† Nathan G. Walcott,‡ and Jeff J. Tollett‡


Sequencing-by-Hybridization (SBH)

Arrayed Primer Extension (APEX)

Figure 1. Hybridization on DNA chips is simply mass action driven, and the target must bear a label. APEX involves a preequilibrium followed by an irreversible enzymatic reaction, both steps of which are sensitive to mismatches.

J. Am. Chem. Soc. **2000,** *122,* 1873−1882

Surface computer II

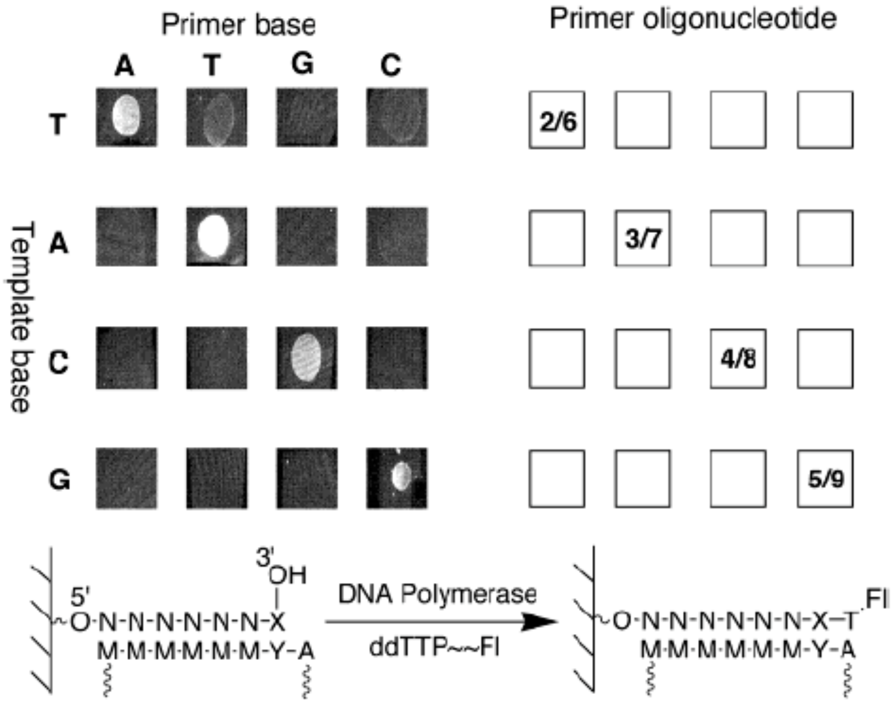


Figure 5. Experiment addressing the fidelity of primer extension. Primers differing only at the X position and templates differing only at the Y position were prepared, and the primers were arrayed in columns. The templates were applied to those primers in rows. Thus, at each priming site, a different combination of X and Y (of the 16 possible) is formed. Only when complementary base pairs are formed is extension signal observed. The sequences of the primers are given Table 1.

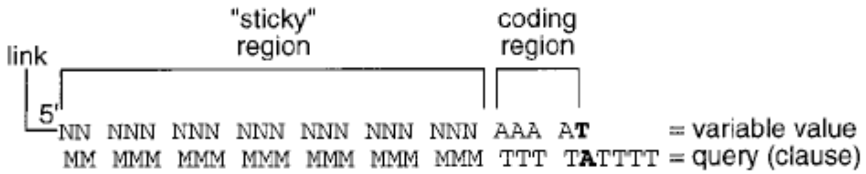
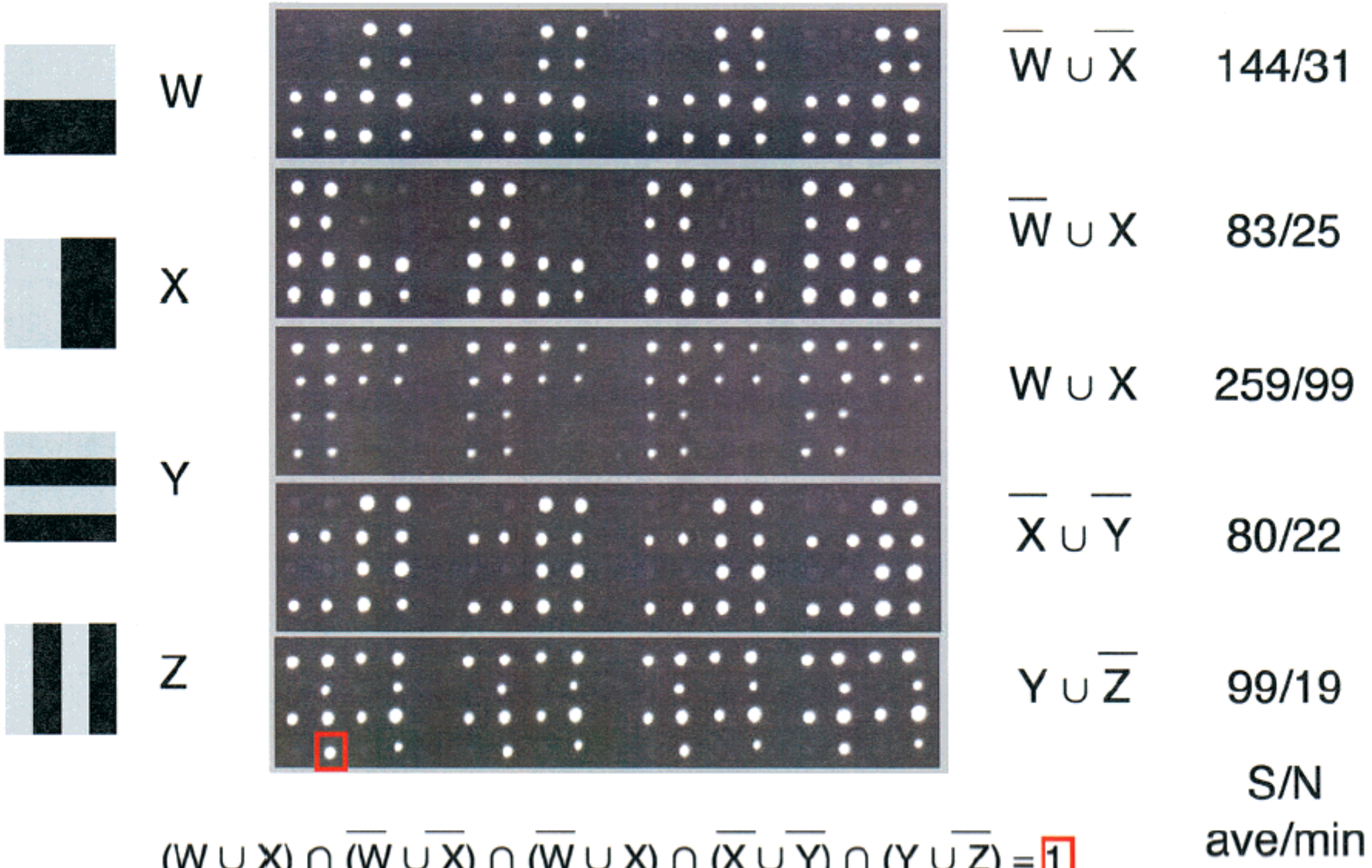

link
"sticky" region
coding region
5'
NN NNN NNN NNN NNN NNN NNN AAA AT
MM MMM MMM MMM MMM MMM MMM TTT TATTTT
= variable value
= query (clause)

Figure 6. Encoding scheme for primers and templates used in APEX computation. A complementary 18–24-nt constant region in each provides a perfect duplex in the primer–template complex. A 5-nt coding region is located at the 3′ end of the primer, where polymerases show high discrimination against mismatches. Complements to this coding region are at the 5′ end of the template. Ts follow this region, leading to the addition of ddATP to the primer strand. Each query or clause requires one to several of these templates.

J. Am. Chem. Soc. **2000,** *122,* 1873–1882

J. Am. Chem. Soc. **2000,** *122,* 1873–1882

Surface computer II

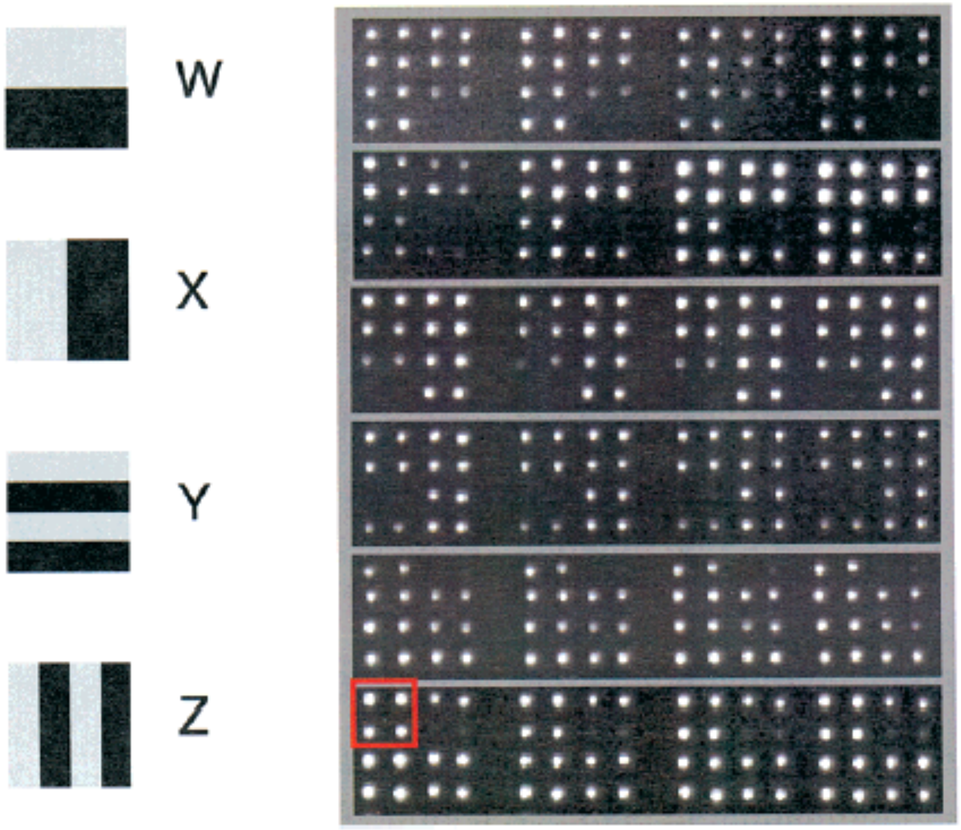

W
X
Y
Z
$\overline{W} \cup \overline{X}$ 144/31
$\overline{W} \cup X$ 83/25
$W \cup X$ 259/99
$\overline{X} \cup \overline{Y}$ 80/22
$Y \cup \overline{Z}$ 99/19
S/N
ave/min

$$(W \cup X) \cap (\overline{W} \cup \overline{X}) \cap (\overline{W} \cup X) \cap (\overline{X} \cup \overline{Y}) \cap (Y \cup \overline{Z}) = \boxed{1}$$

Figure 9. Four-bit SAT. An array of DNA primers was prepared in quadruplicate from eight oligonucleotides, **1**–**4** and **9**–**12**. The key on the left shows the distribution of primers for each variable (grey = 1, black = 0). Four primers were pooled for each array element, yielding 16 sites with unique combinations of the TRUE and FALSE state primers for four variables. Approximately 10 nL of 100 μM oligonucleotides solution was deposited per spot, with spot diameters of 300 μm spaced on 1-mm centers. After one APEX for each clause of the SAT, using two templates (as shown to the right of the images) selected from among **5**–**8** and **13**–**16**, measurements of S/N ratio were made at each spot in the array. The numbers given represent the ratios of the signal at TRUE primer extension sites to FALSE sites. The average S/N ratio (#:1) observed on all four arrays and the minimum S/N ratio on one array are given. The unique solution, boxed in red, is ($w = 0$, $x = 1$, $y = 0$, $z = 0$).

Surface computer II

J. Am. Chem. Soc. **2000,** *122,* 1873−1882



	S/N ave/min
$W \cup X \cup Y$	188/68
$W \cup X \cup \overline{Y}$	76/4
$W \cup \overline{X} \cup Y$	63/7
$W \cup \overline{X} \cup \overline{Y}$	51/22
$\overline{W} \cup X \cup Y$	27/8
$\overline{W} \cup X \cup \overline{Y}$	21/10

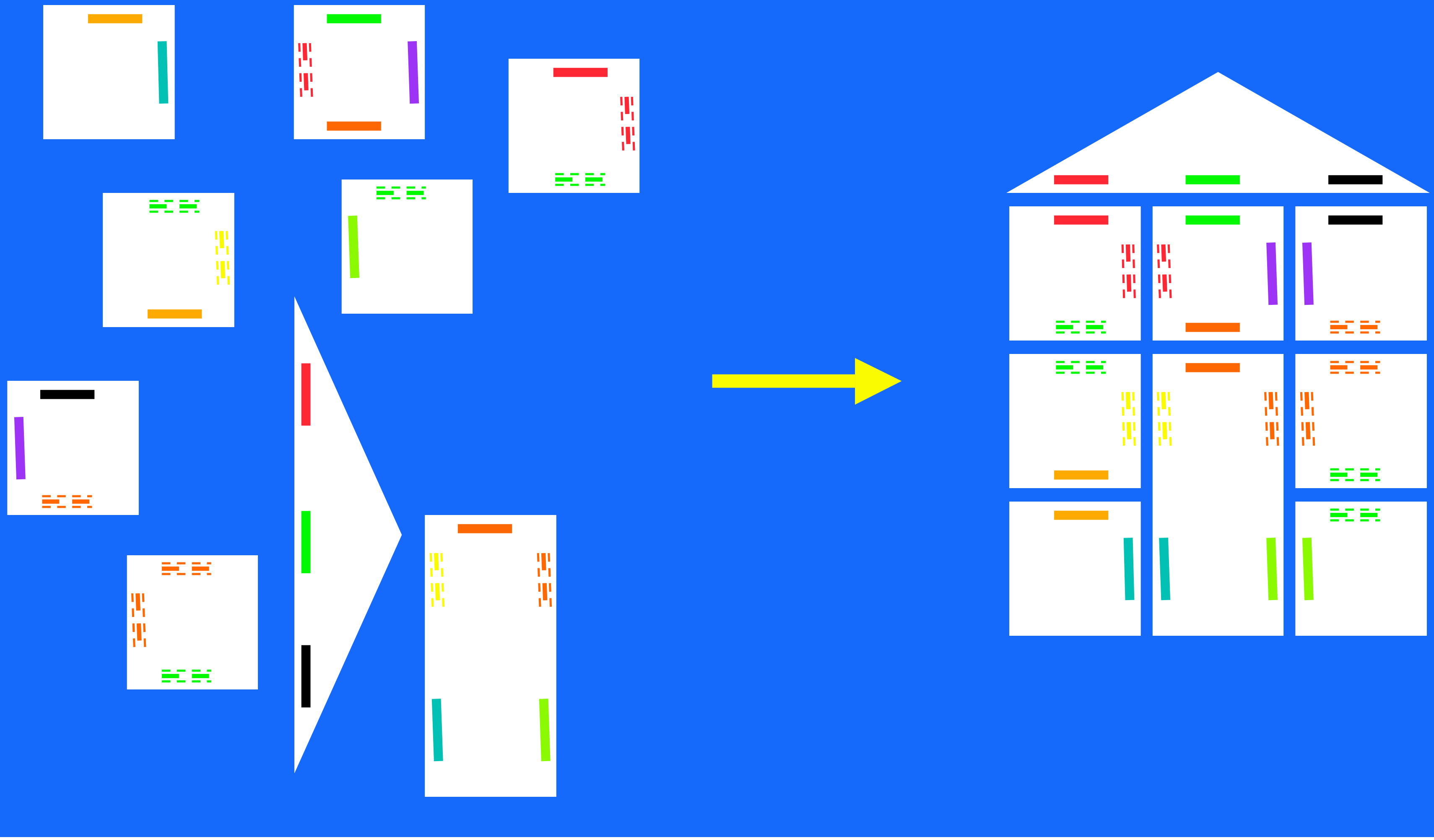
$$(W \cup X \cup Y) \cap (W \cup X \cup \overline{Y}) \cap (W \cup \overline{X} \cup Y) \cap (W \cup \overline{X} \cup \overline{Y}) \cap (\overline{W} \cup X \cup Y) \cap (\overline{W} \cup X \cup \overline{Y}) = \boxed{1}$$

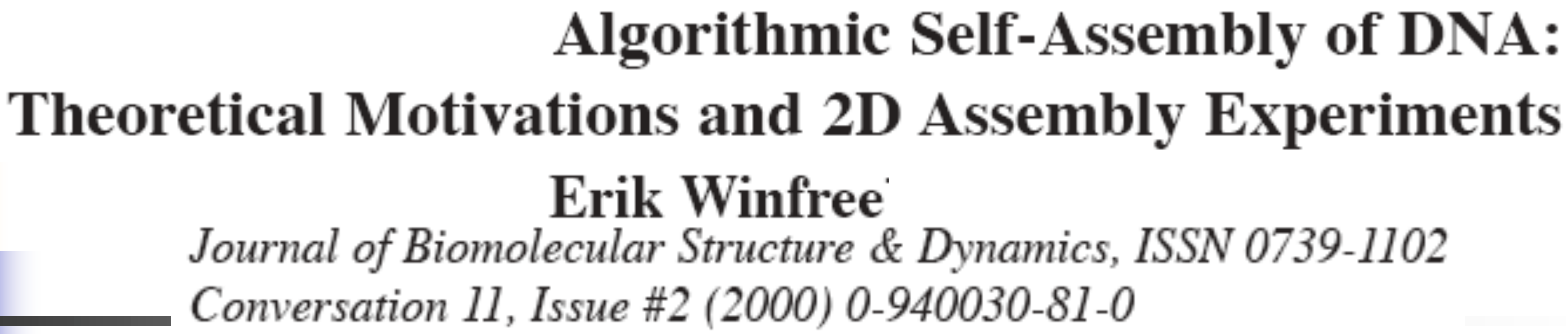
Figure 10. Three-bit 3SAT. Following a protocol similar to that in Figure 9, six slides were incubated using three templates specific for each clause, as shown to the right of the images. Each clause eliminates two spots from the set of 16, so that the commonly labeled elements represent the solution boxed in red ($w = 1$, $x = 1$, $y = 0 \cup 1$, $z = 0 \cup 1$). In this 3SAT, there are no constraints on z, so both values satisfy the expression. The S/N data are tabulated to the right of each image, as in Figure 9. The lower minimum S/N numbers on several of the conditions are likely due to carry-over during the array printing process. The minimum value of 68 for the first clause is more representative of usual APEX data because, in this case, the FALSE (unlabeled) spots are the first to be printed.

DNA-based Computing

- Adleman - Hamiltonian Path (7 node)
- Biochemical manipulation
- Sticker model, RNA, Whiplash PCR
- Surface-based computation
- Algorithmic self-assembly
- Autonomous systems

Construction with “Smart Bricks”

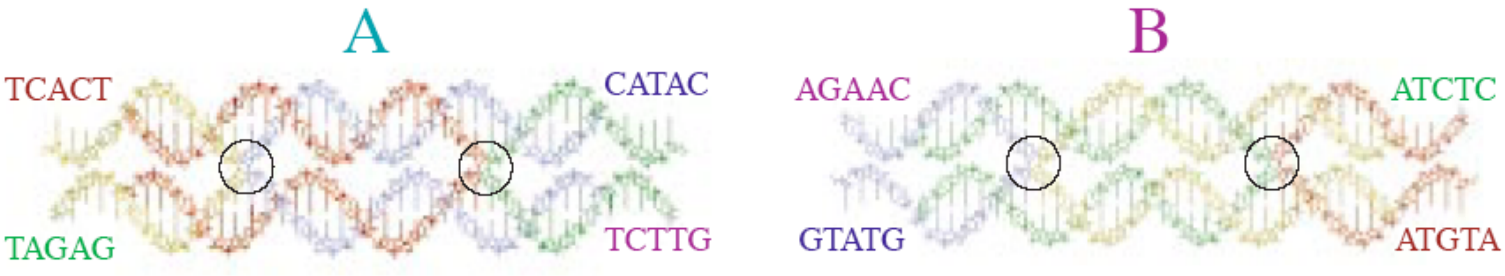




Algorithmic Self-Assembly of DNA: Theoretical Motivations and 2D Assembly Experiments

Erik Winfree

Journal of Biomolecular Structure & Dynamics, ISSN 0739-1102
Conversation 11, Issue #2 (2000) 0-940030-81-0



- Annealing alone is capable of universal computation.
- Combine Wang tiling problems with Seeman DNA structures.
  - Wang tiling: Given a set of polygons, cover the entire plane without gaps. Given colored tile edges, match neighboring edge colors.
  - Tiling patterns have been shown which simulate a single-tape Turing machine.

Algorithmic SA

Journal of Biomolecular Structure & Dynamics, ISSN 0739-1102
Conversation 11, Issue #2 (2000) 0-940030-81-0

- Simple tile sets.
- Simple rule sets.
- Simple periodic structures produced.

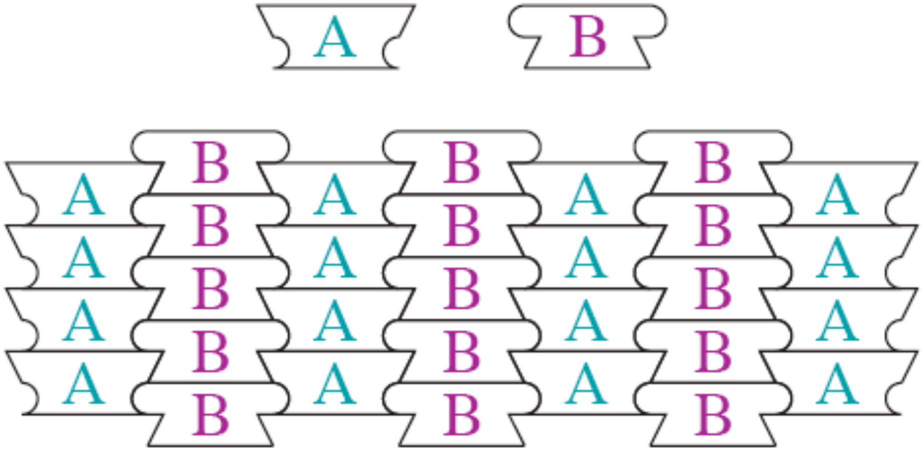


Figure 1: A system of 2 tiles that form a periodic striped lattice.

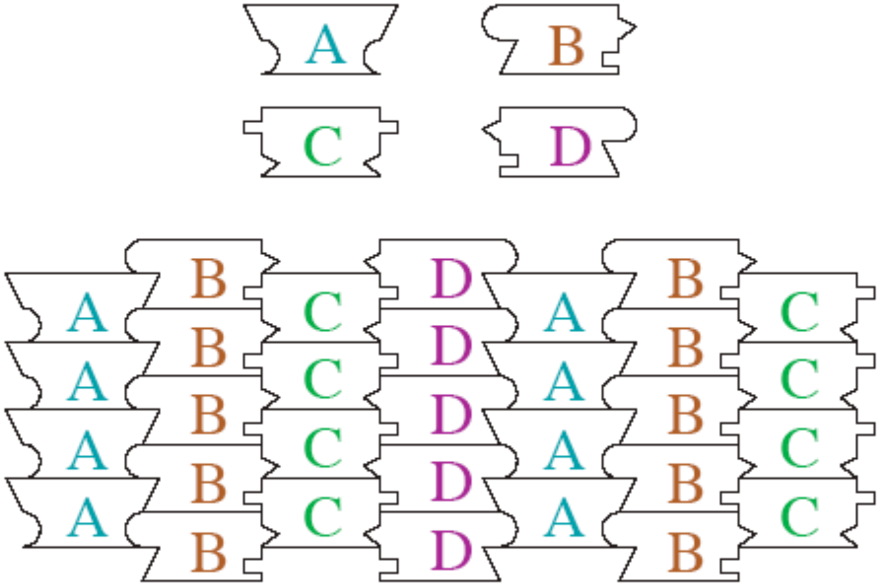


Figure 2: A system of 4 tiles that form a periodic striped lattice.

Algorithmic SA

- Small tile sets.
- Encoded rules on edge symbols.
- Aperiodic structure produced. Algorithmic construction of subsequent binding sites.

Journal of Biomolecular Structure & Dynamics, ISSN 0739-1102
Conversation 11, Issue #2 (2000) 0-940030-81-0

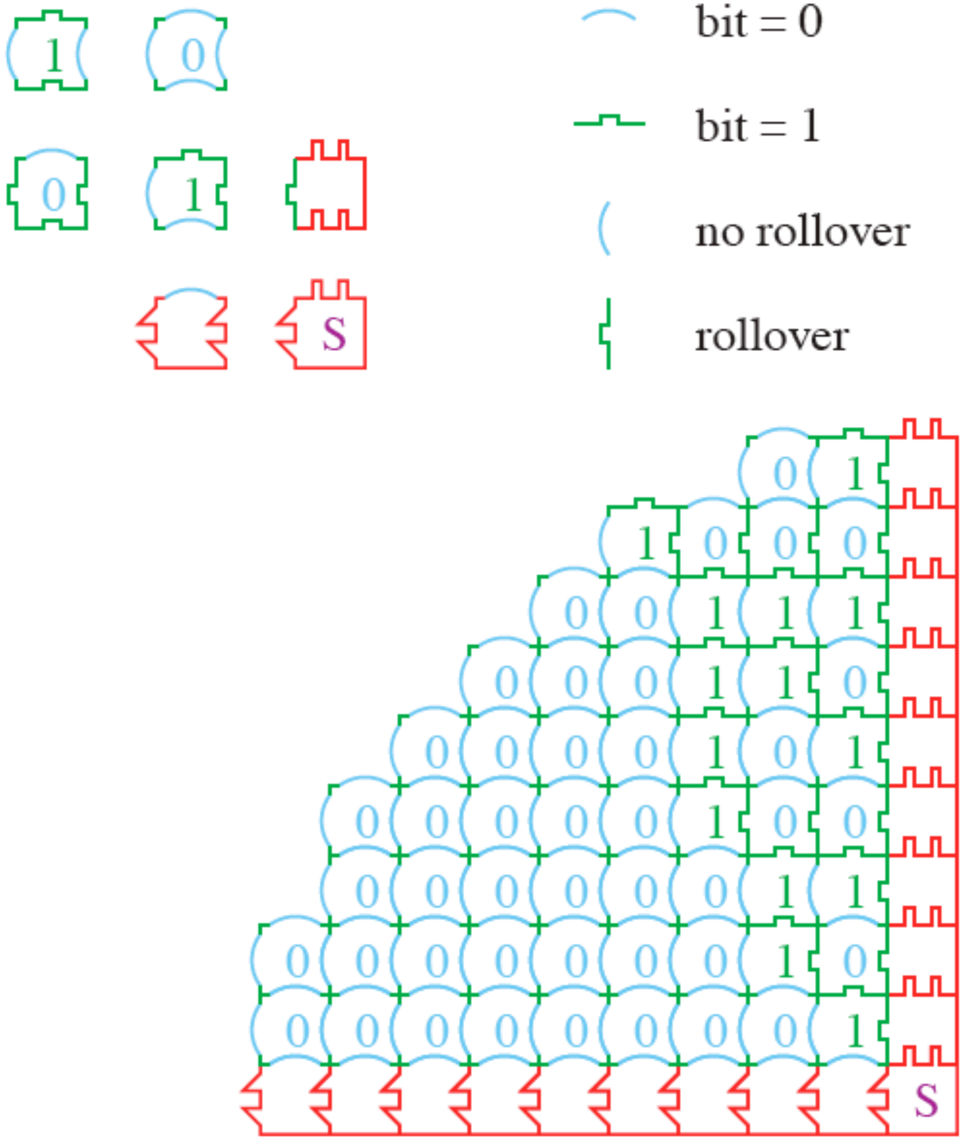
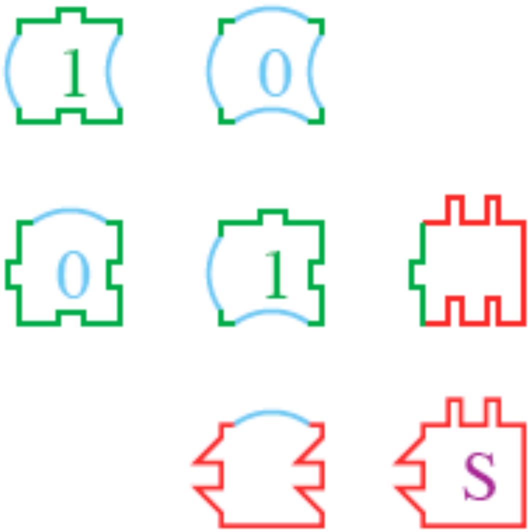
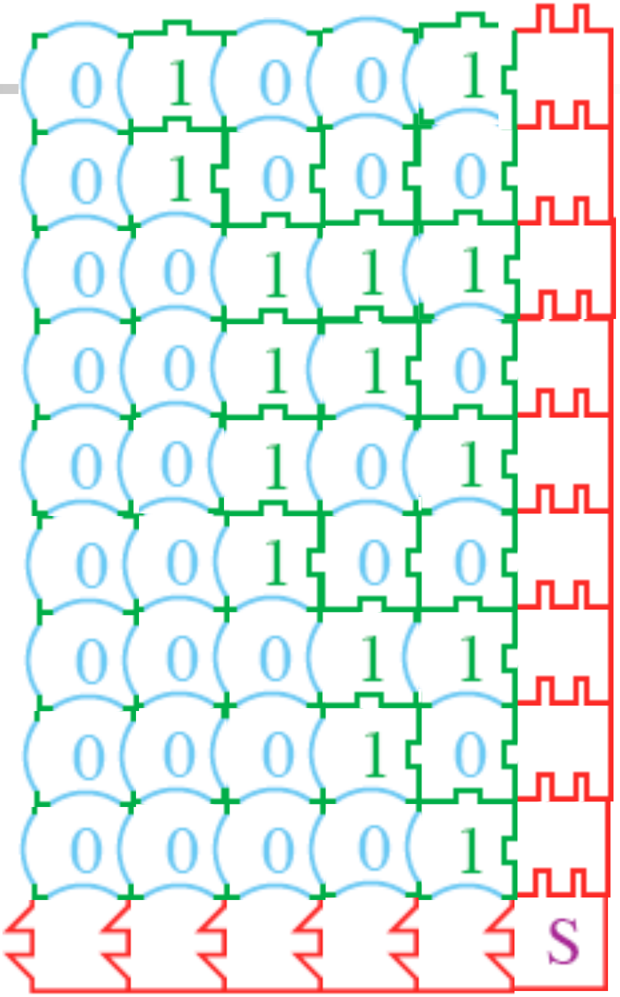


Figure 3: A system of 3 input tiles and 4 rule tiles that form an aperiodic tiling. The rows in the tiling are the consecutive integers, represented in binary.

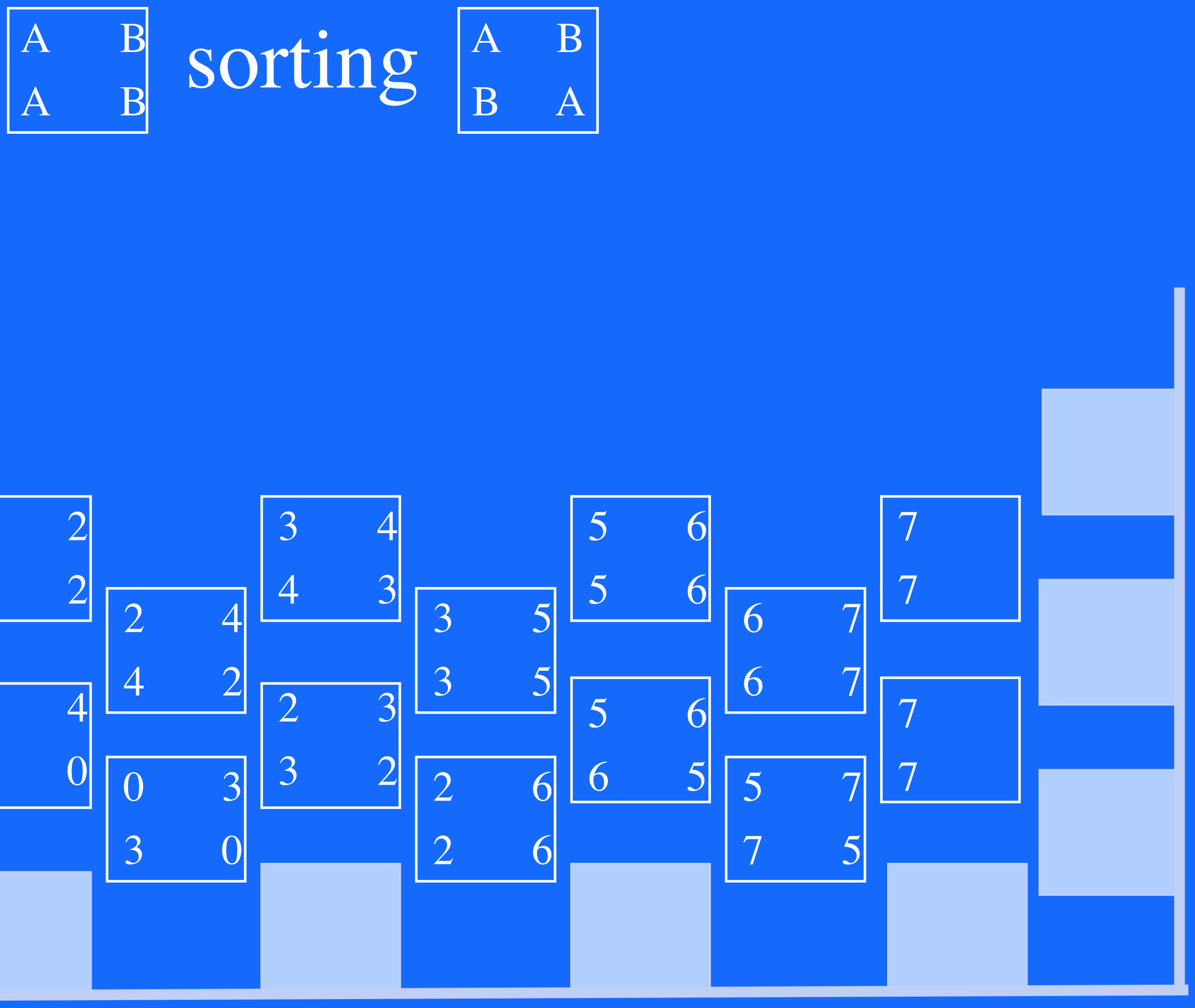
Algorithmic SA

Journal of Biomolecular Structure & Dynamics, ISSN 0739-1102
Conversation 11, Issue #2 (2000) 0-940030-81-0



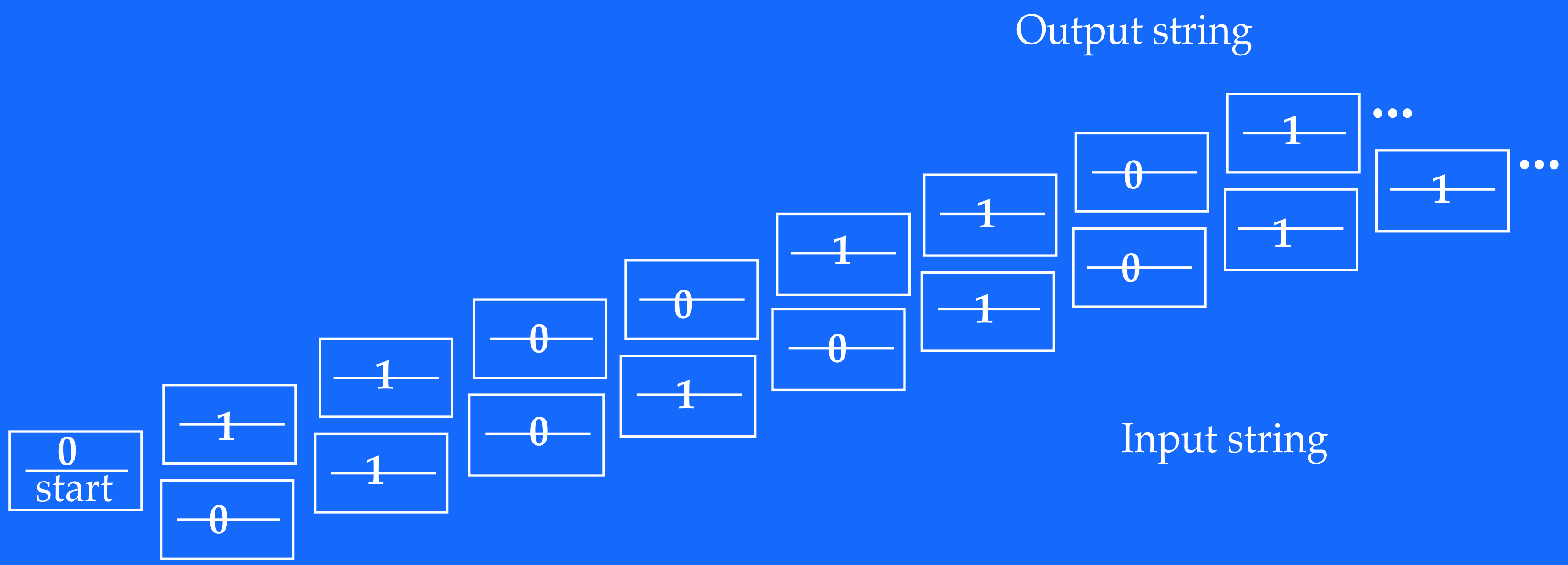


Computation with “Smart Bricks”



Contingency Table and Tiling Solution for Boolean xOr

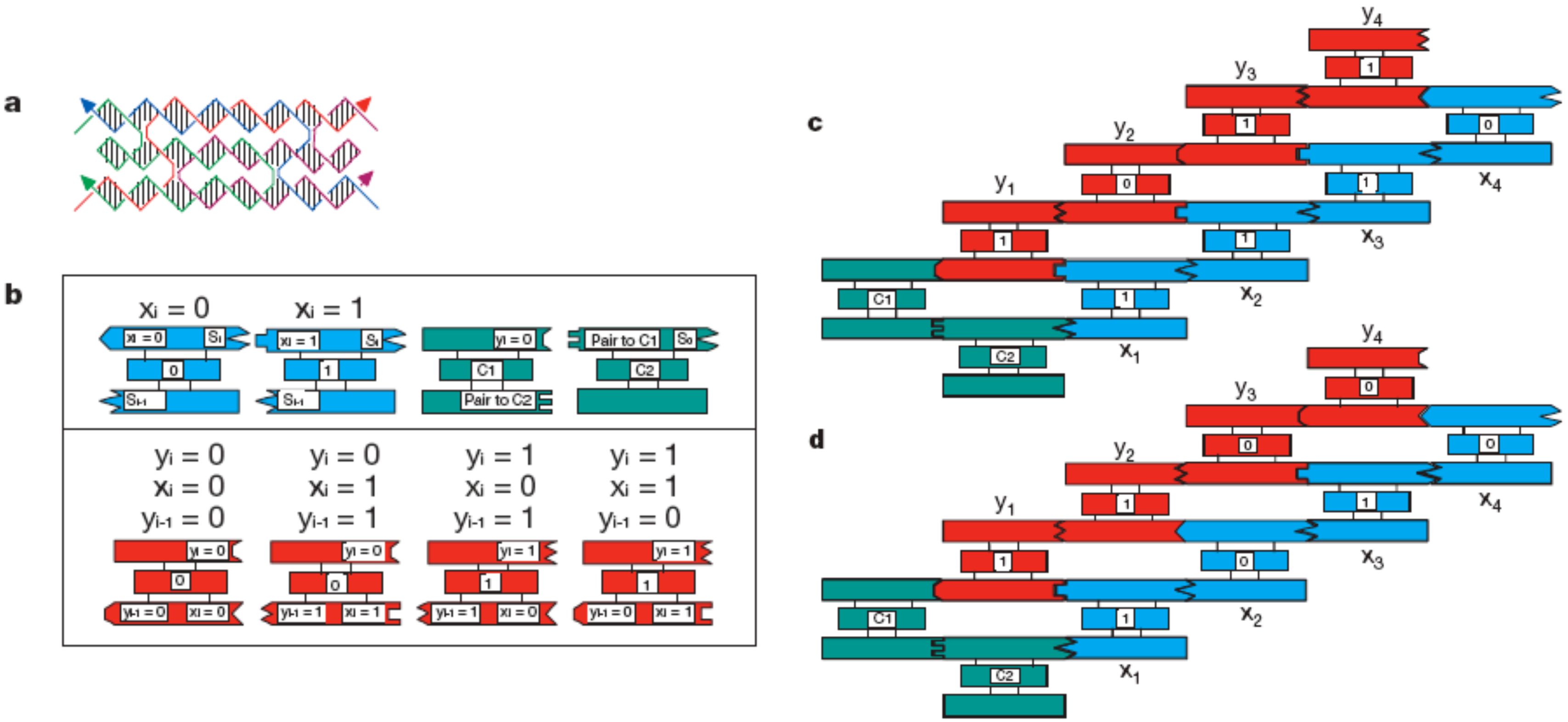
IN	OUT
0 0	0
0 1	1
1 0	1
1 1	0



Logical computation using algorithmic self-assembly of DNA triple-crossover molecules

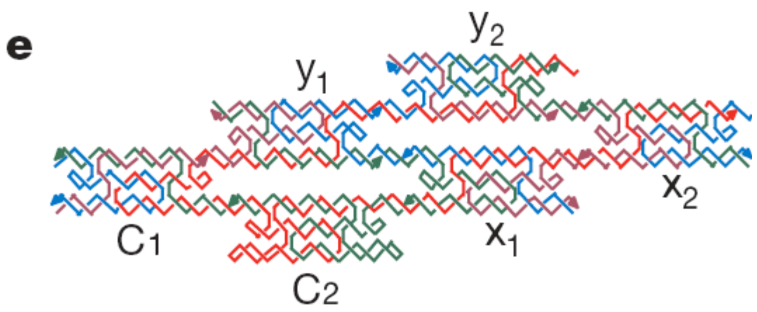
NATURE | VOL 407 | 28 SEPTEMBER 2000

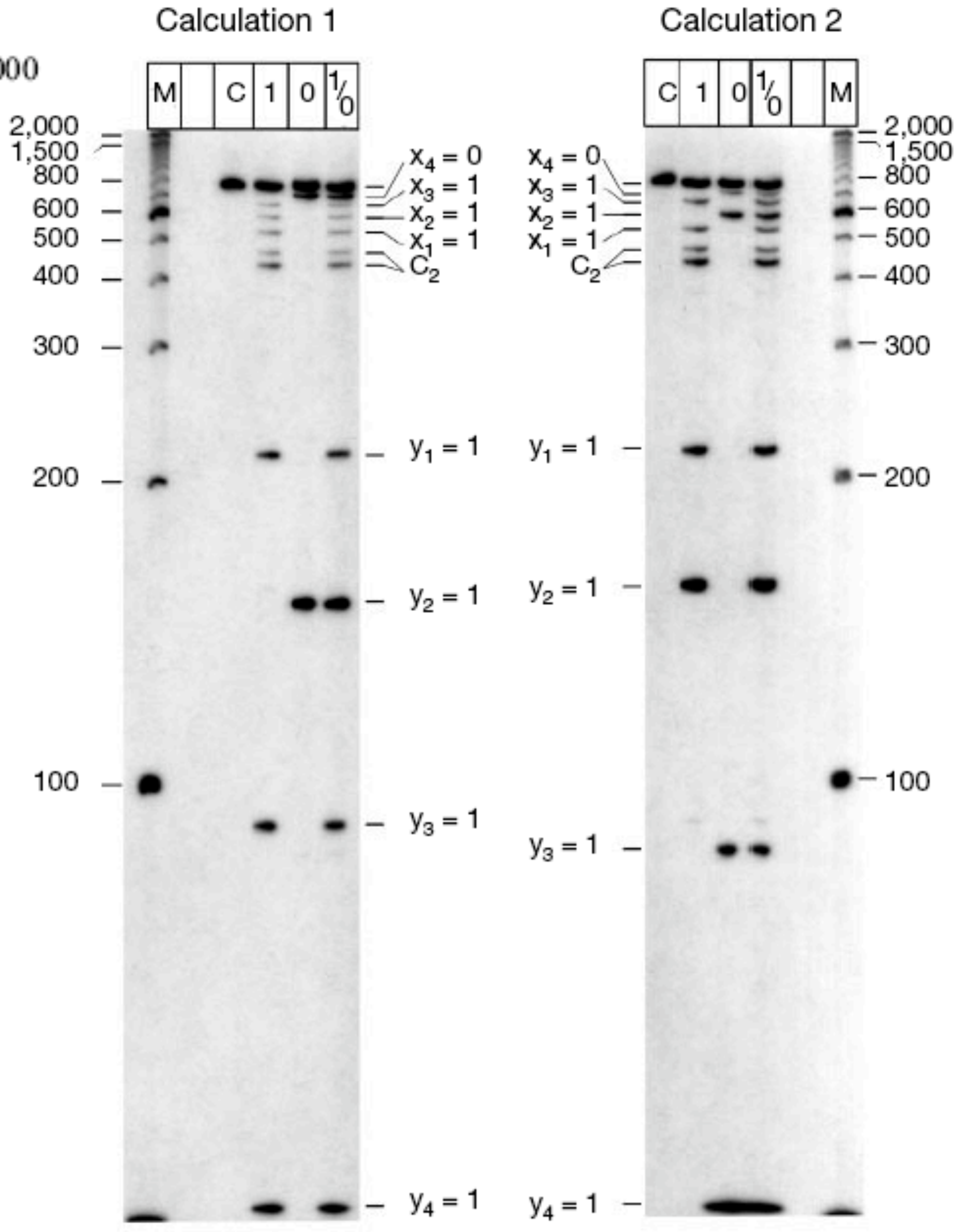
Chengde Mao*, Thomas H. LaBean†, John H. Reif† & Nadrian C. Seeman *



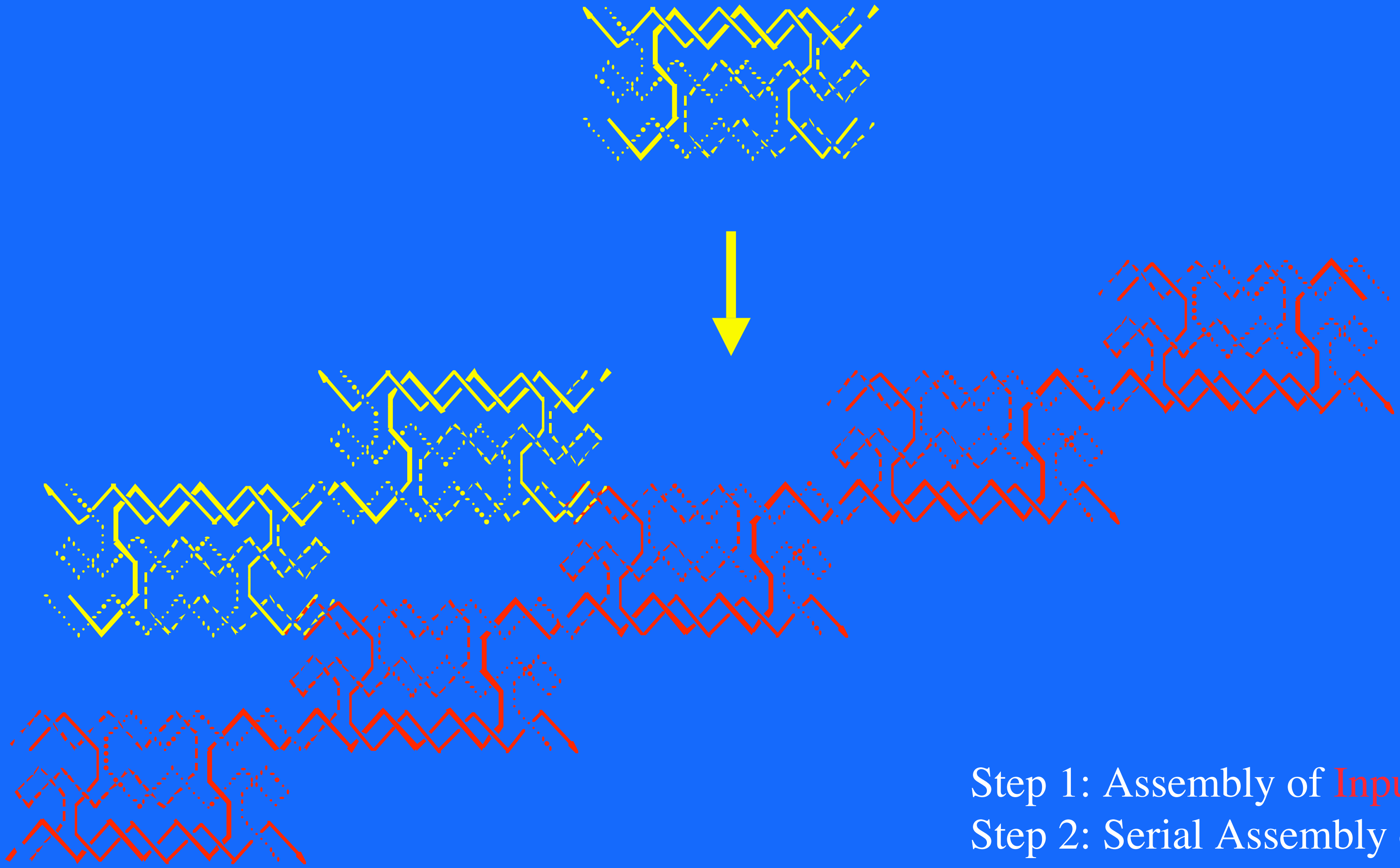
TAO XOR

NATURE | VOL 407 | 28 SEPTEMBER 2000





Locally Serial, Globally Parallel 2D Assembly



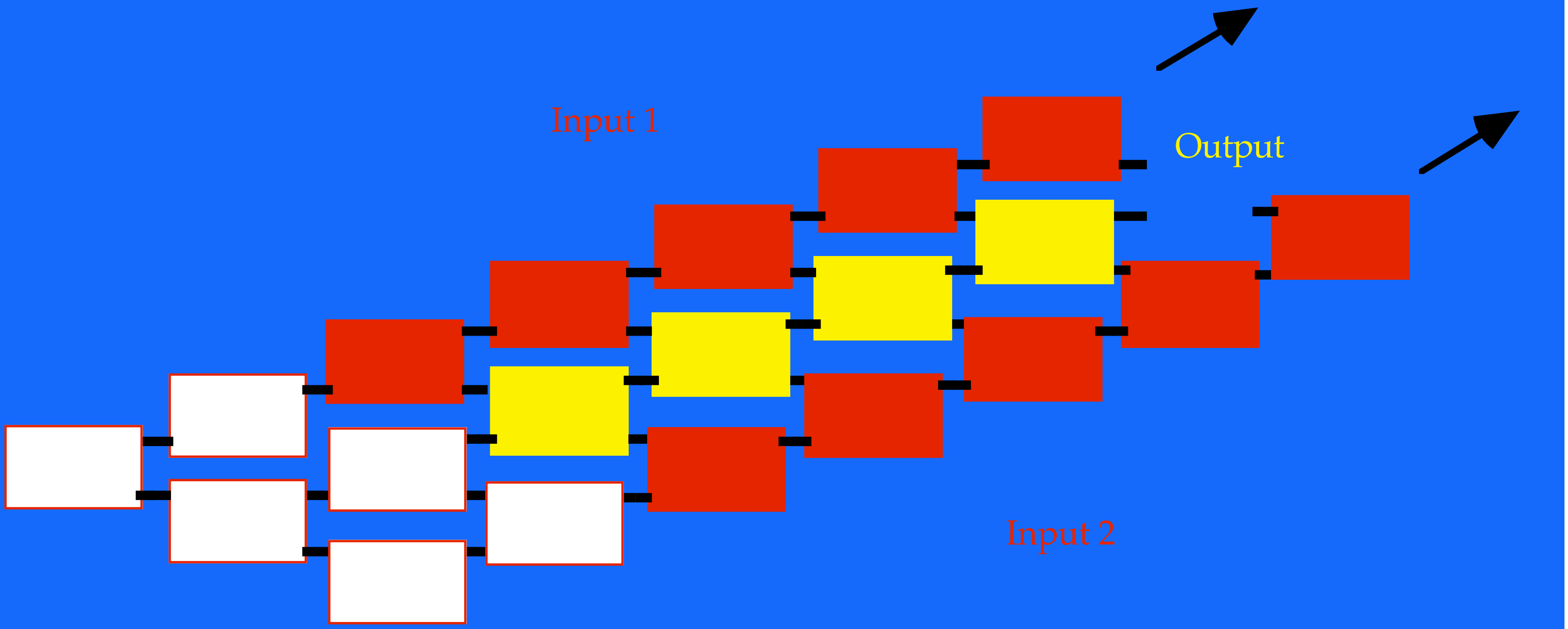
Step 1: Assembly of Input

Step 2: Serial Assembly of Output

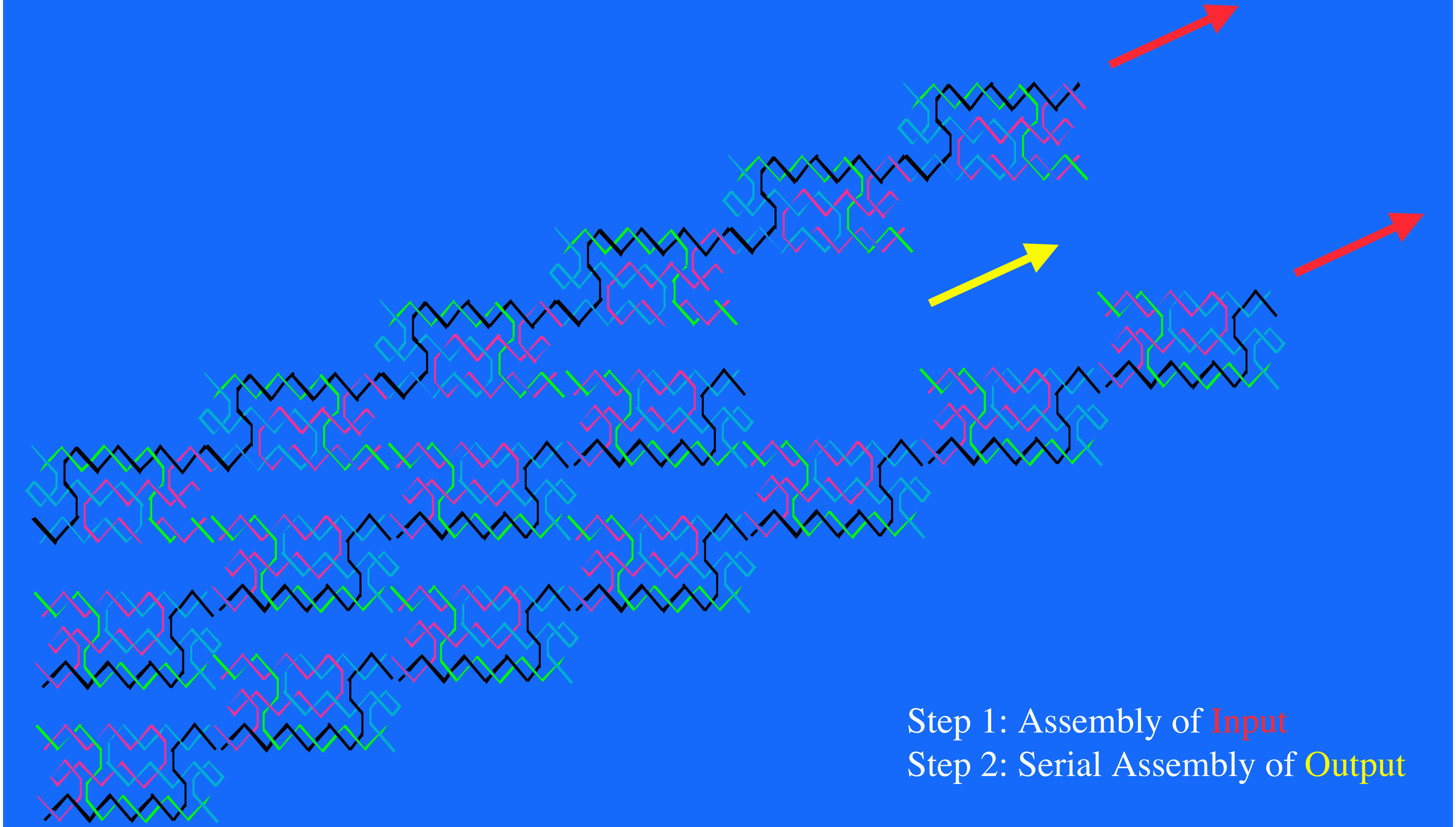
Advantages of Biomolecular Computation

- **Ultra Scale:** each “processor” is a molecule.
- **Massively Parallel:** number of elements could be 10^{18} to 10^{20}
- **High Speed:** perhaps 10^{15} operations per second.
- **Low Energy:** example calculation ~10^{-19} Joules/op.
  electronic computers ~10^{-9} Joules/op.
- **Existing Biotechnology:** well tested recombinant DNA techniques.

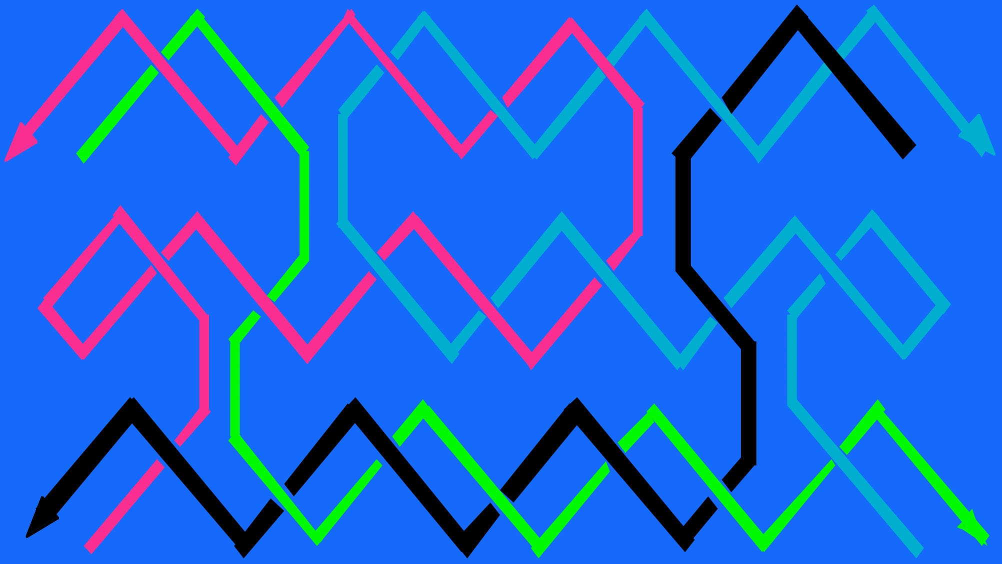
Input Frame and Output Assembly for Parallel Addition



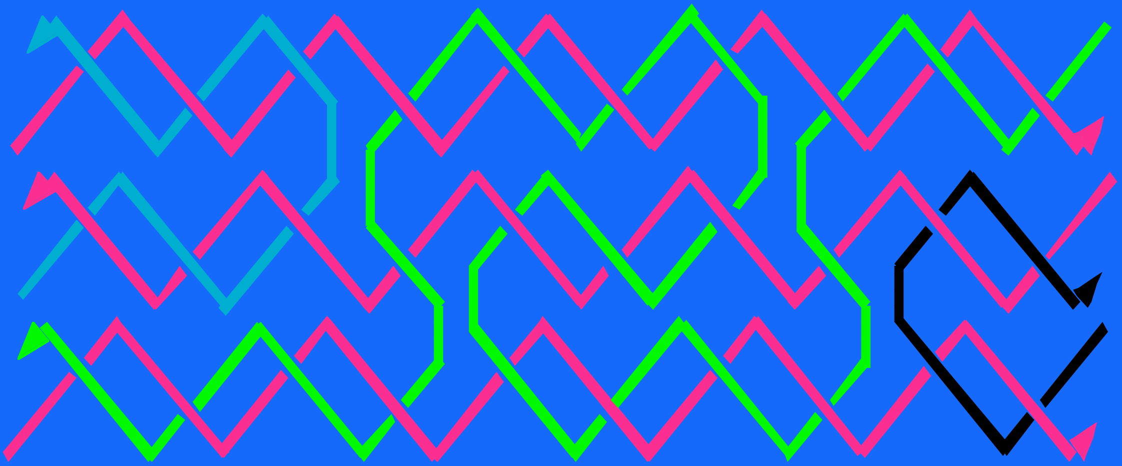
Locally Serial, Globally Parallel 2D Assembly



Step 1: Assembly of Input
Step 2: Serial Assembly of Output

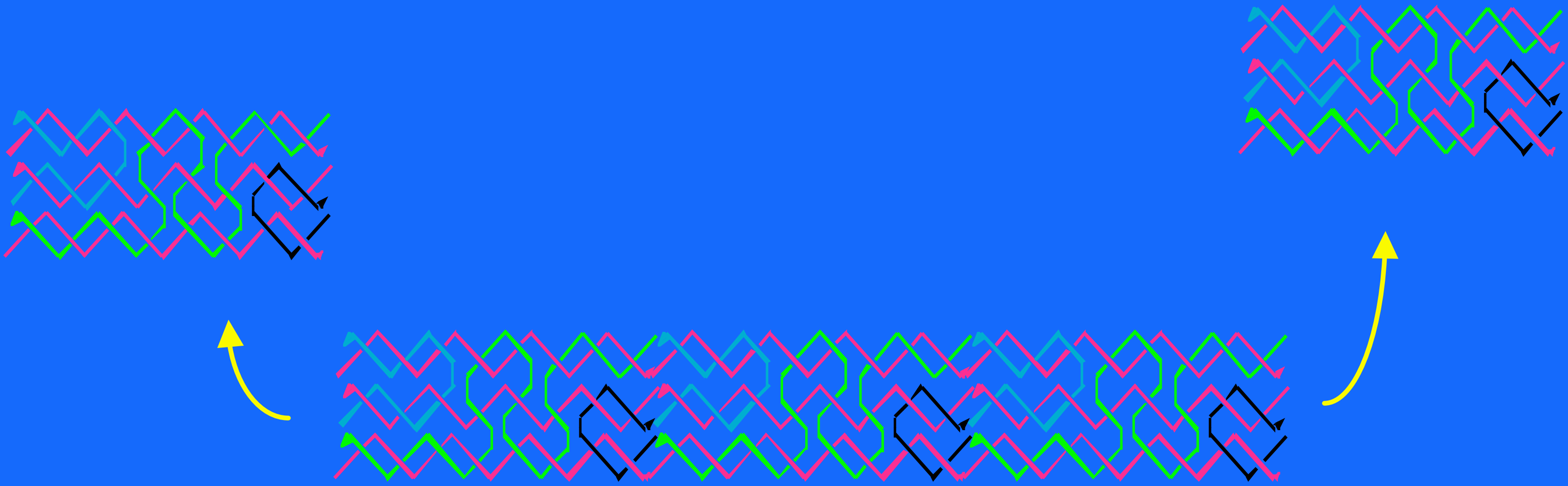


TAO35



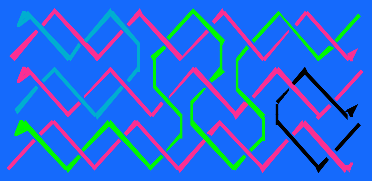
TAE 44

Global and Local Parallelism by Linear Assembly

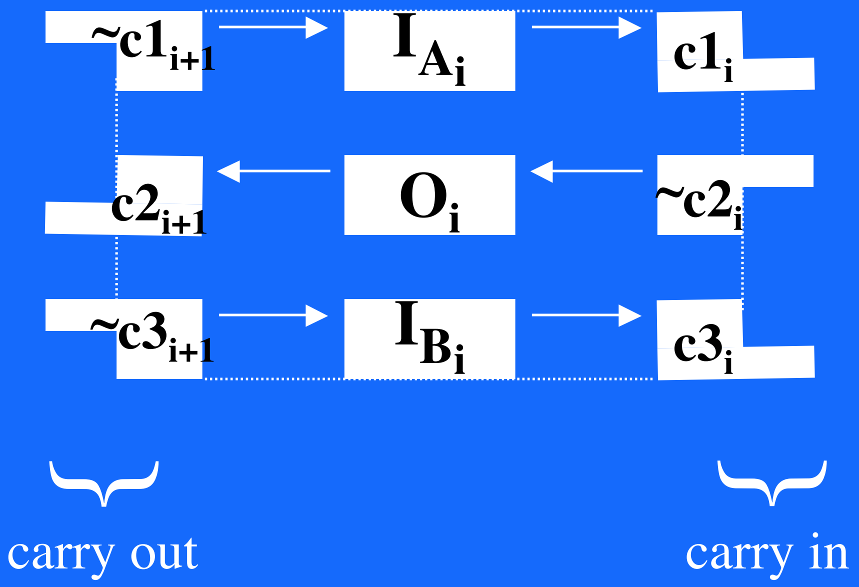


Simultaneous, parallel assembly of input and output strands.

“String Tile” Addition



tile	c_i	I_{A_i}	I_{B_i}	O_i	c_{i+1}
1	0	0	0	0	0
2	0	0	1	1	0
3	0	1	0	1	0
4	0	1	1	0	1
5	1	0	0	1	0
6	1	0	1	0	1
7	1	1	0	0	1
8	1	1	1	1	1



$\sim c1_{i+1}$ I_{A_i} $c1_i$

$c2_{i+1}$ O_i $\sim c2_i$

$\sim c3_{i+1}$ I_{B_i} $c3_i$

carry out

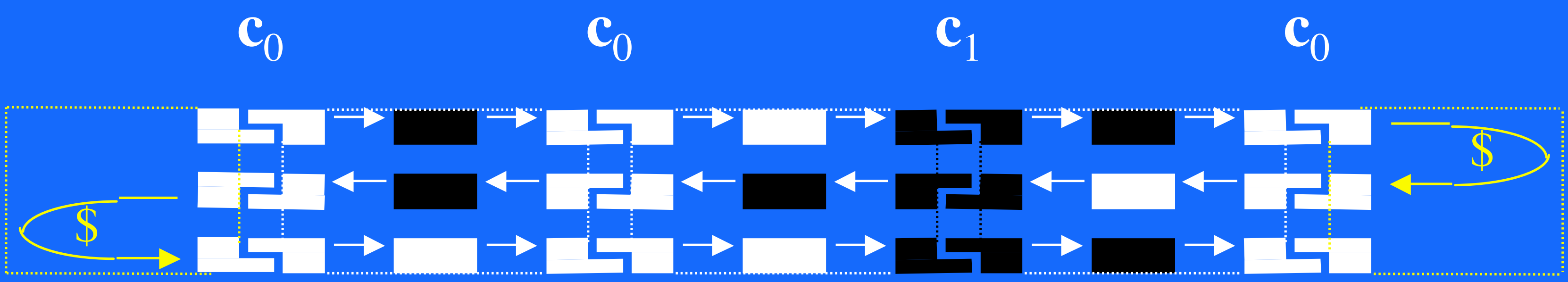
carry in

8 tile types needed, indexed according to each row in the truth table.

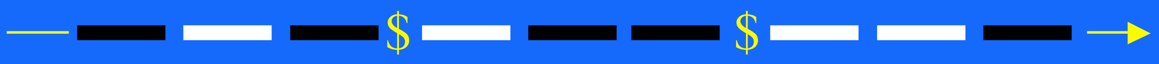
“String Tile” Addition. Example.

□ 0

■ 1



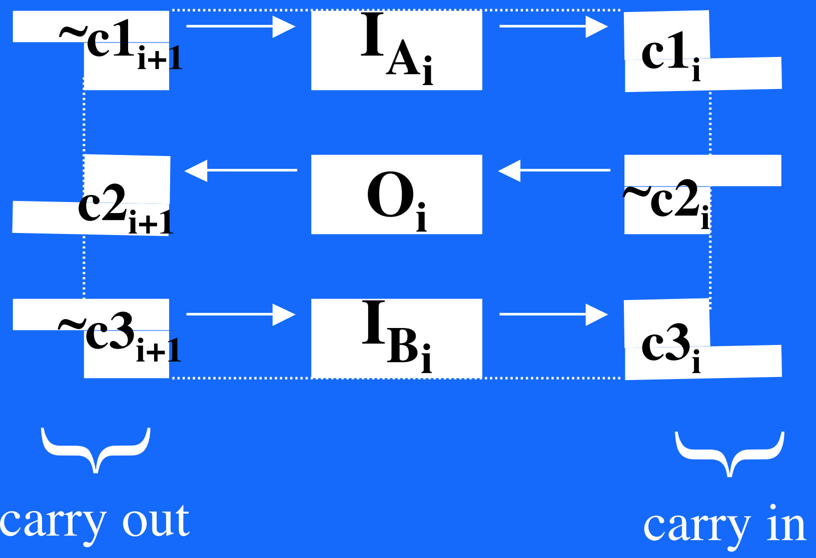
- Anneal; Ligate
- Purify reporter strand



$1\quad 0\quad 1\quad \$\quad 0\quad 1\quad 1\quad \$\quad 0\quad 0\quad 1\quad = \quad I_A\ \$\ O^R\ \$\ I_B$

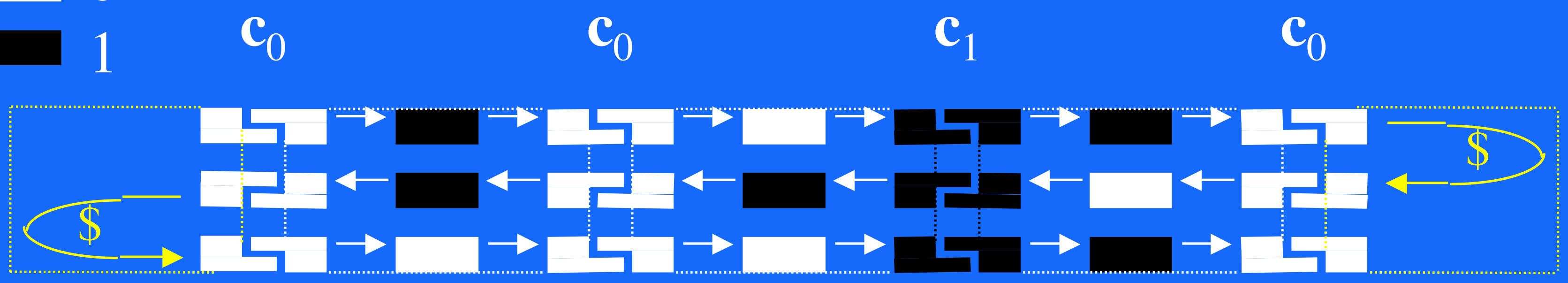
$$\begin{array}{r}101\\ +\underline{001}\\ 110\end{array}$$

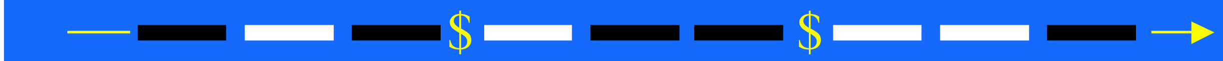
tile	c_i	I_{A_i}	I_{B_i}	O_i	c_{i+1}
1	0	0	0	0	0
2	0	0	1	1	0
3	0	1	0	1	0
4	0	1	1	0	1
5	1	0	0	1	0
6	1	0	1	0	1
7	1	1	0	0	1
8	1	1	1	1	1



□ 0

■ 1





1 0 1 $ 0 1 1 $ 0 0 1 = I_A $ O^R $ I_B

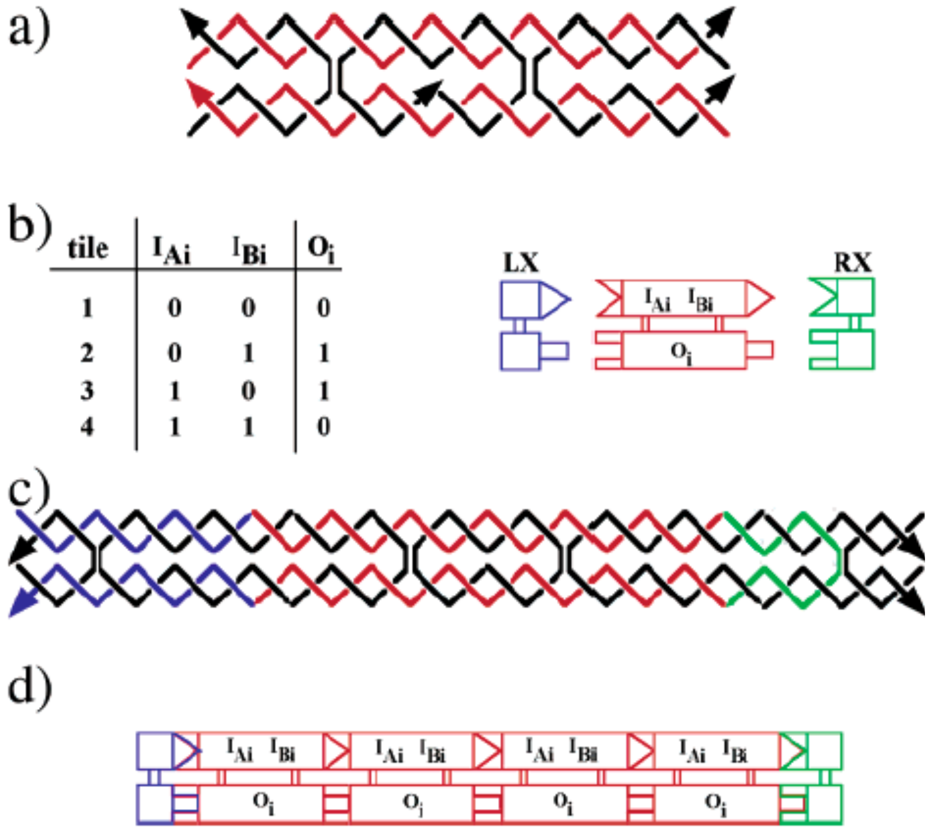
$$\begin{array}{r} 101 \\ +\underline{001} \\ 110 \end{array}$$

Parallel Molecular Computations of Pairwise Exclusive-Or (XOR) Using DNA "String Tile" Self-Assembly

Hao Yan,* Liping Feng, Thomas H. LaBean, and John H. Reif*

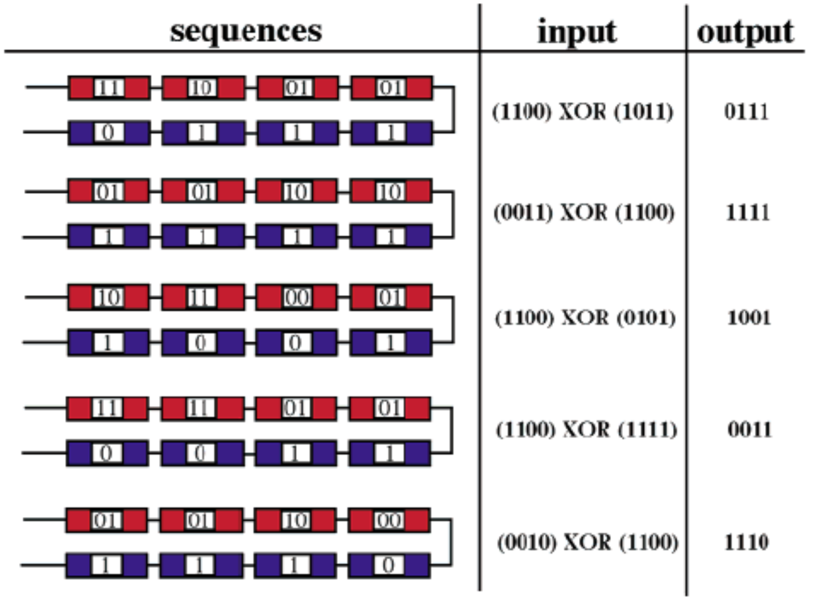
Department of Computer Science, Duke University, Durham, North Carolina 27708

J. AM. CHEM. SOC. 2003, *125*, 14246−14247



tile	I_{Ai}	I_{Bi}	O_i
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Figure 1. Parallel computing of pairwise XOR by self-assembly of DNA tiles.



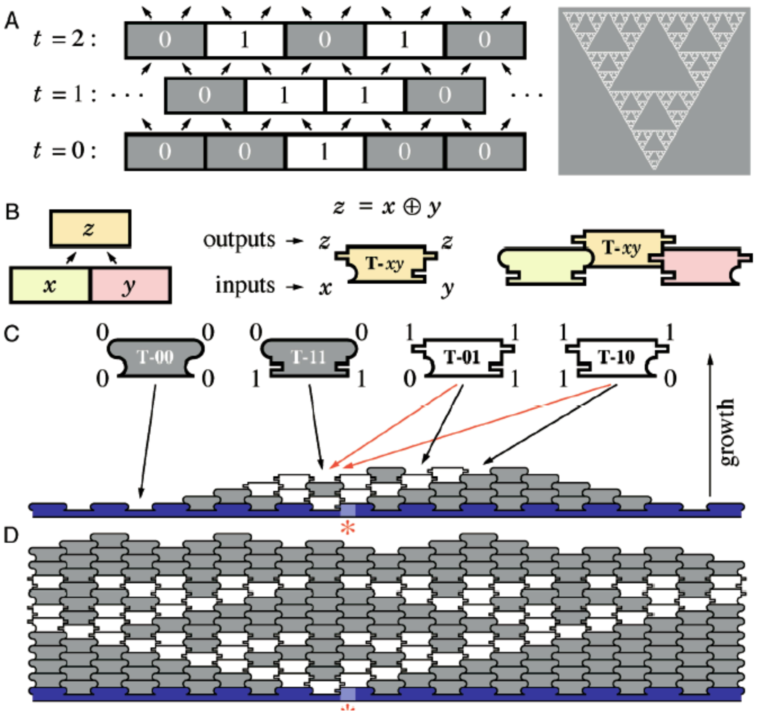
sequences	input	output
11 10 01 01 0 1 1 1	(1100) XOR (1011)	0111
01 01 10 10 1 1 1 1	(0011) XOR (1100)	1111
10 11 00 01 1 0 0 1	(1100) XOR (0101)	1001
11 11 01 01 0 0 1 1	(1100) XOR (1111)	0011
01 01 10 00 1 1 1 0	(0010) XOR (1100)	1110

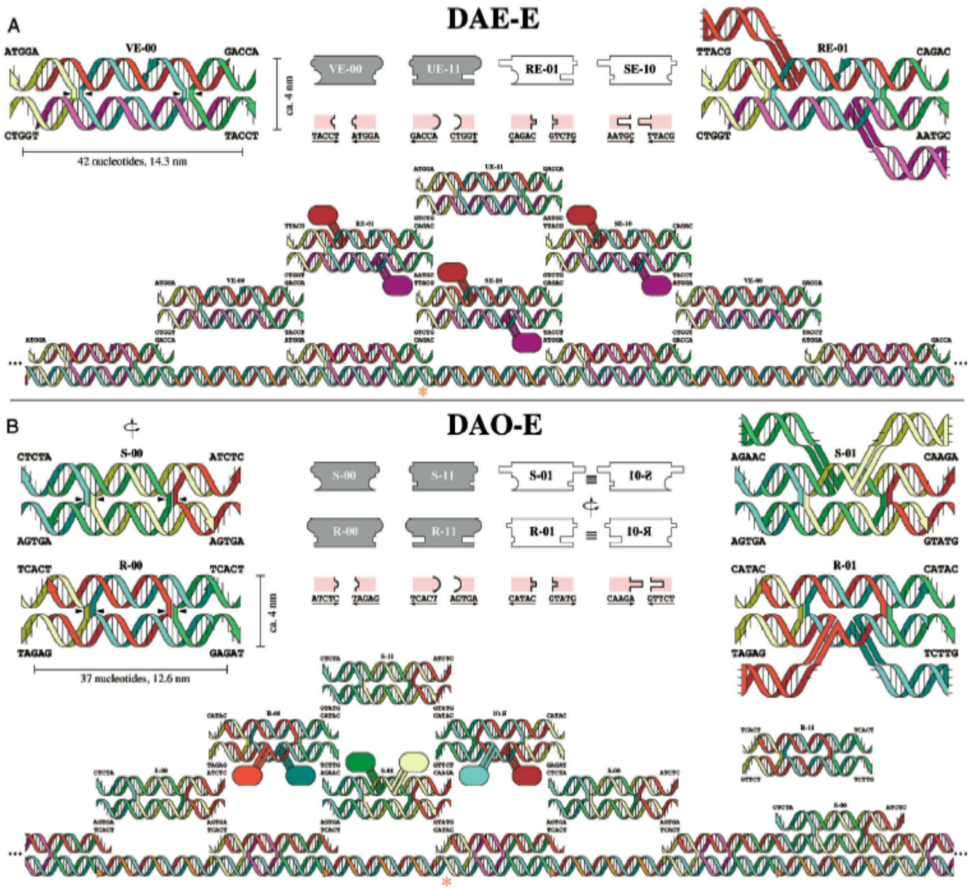
Figure 2. Simplified representation of the sequencing readout for parallel computation of pairwise XOR of "string tile" self-assembly.

Algorithmic Self-Assembly of DNA Sierpinski Triangles

PLoS Biol 2(12): e424.

Paul W. K. Rothemund[1,2], Nick Papadakis[2], Erik Winfree[1,2*]

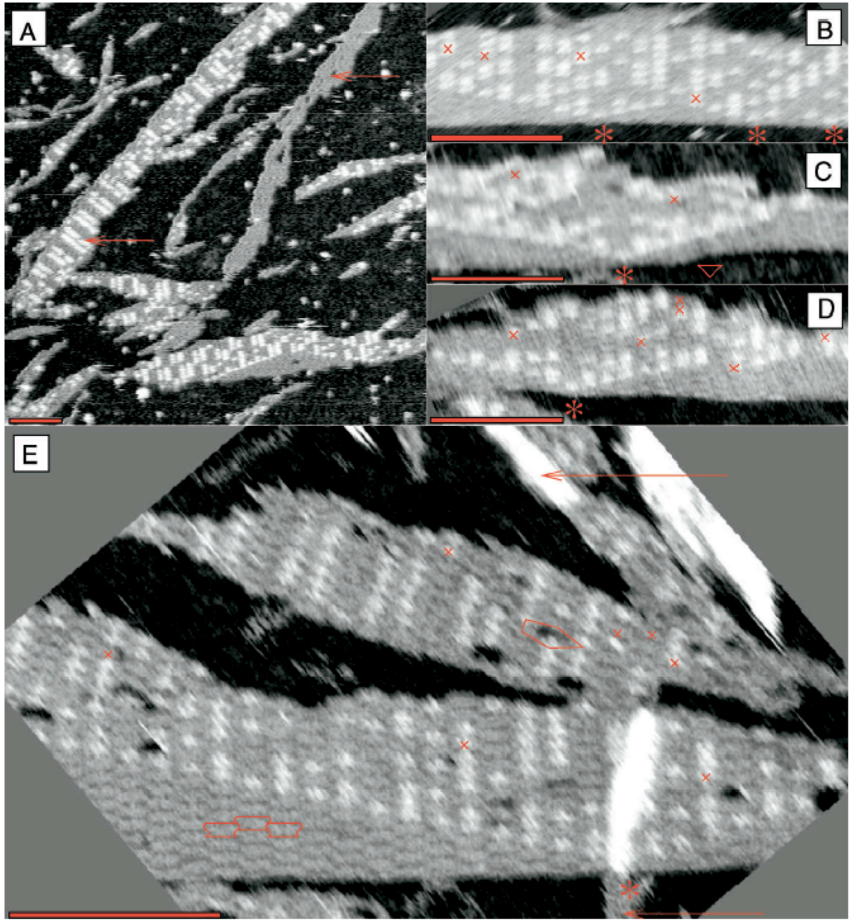




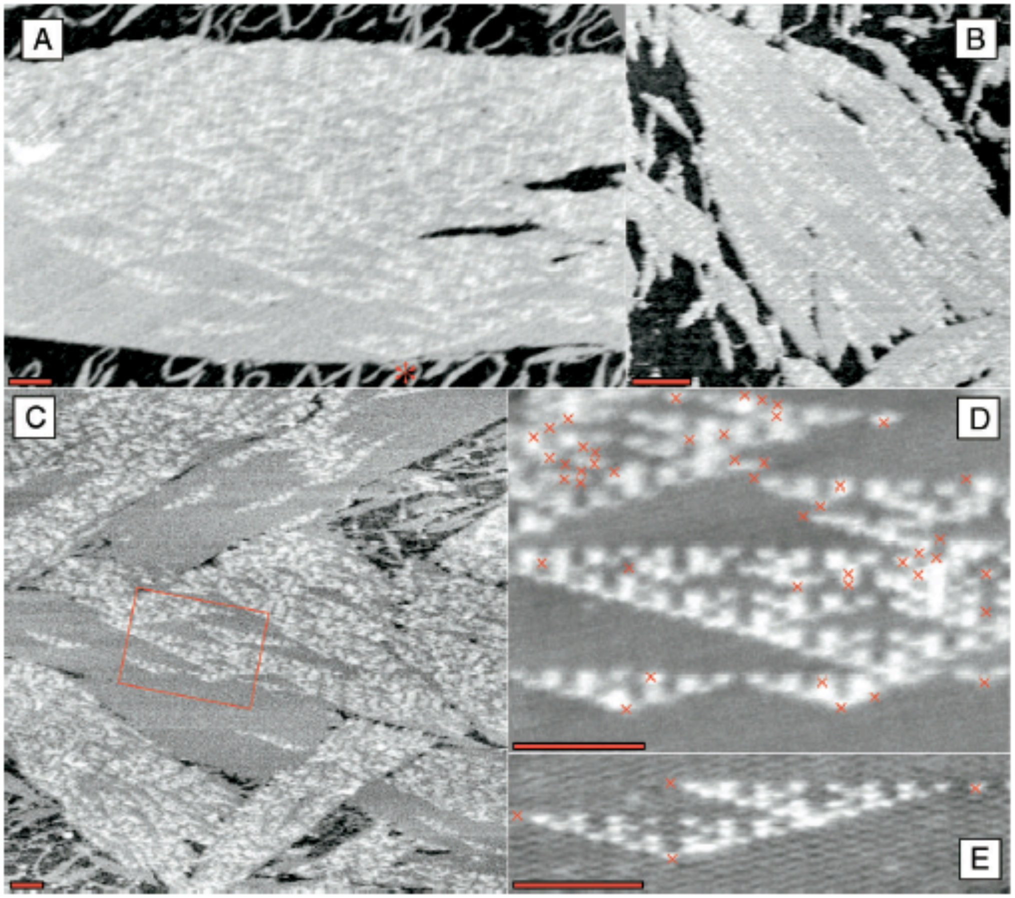
Algorithmic Self-Assembly of DNA Sierpinski Triangles

PLoS Biol 2(12): e424.

Paul W. K. Rothemund[1,2], Nick Papadakis[2], Erik Winfree[1,2*]



DAE-E



DAO-E

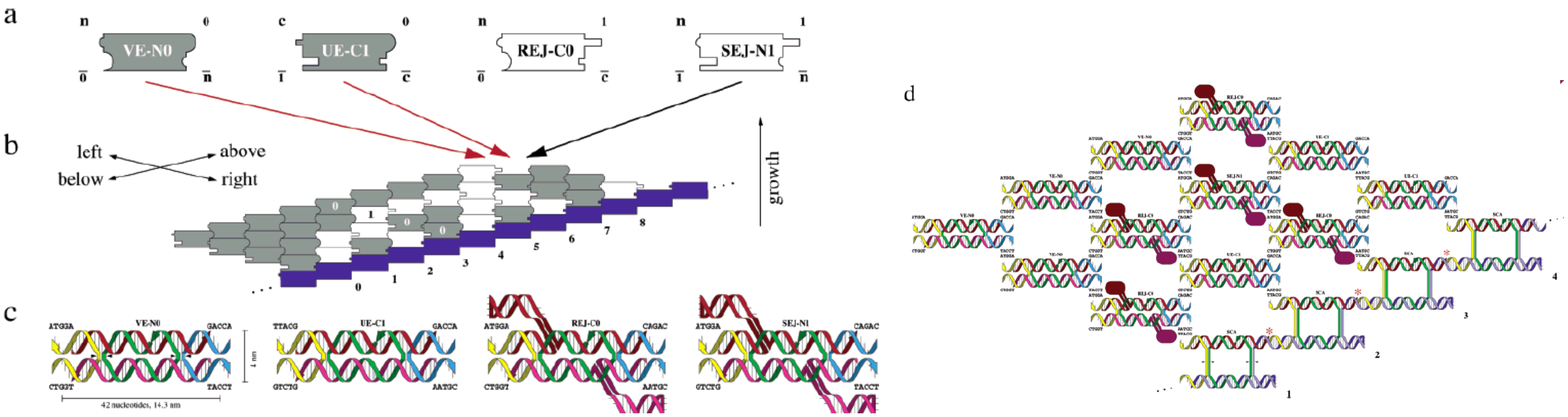
Two Computational Primitives for Algorithmic Self-Assembly: Copying and Counting

Robert D. Barish,† Paul W. K. Rothemund,‡ and Erik Winfree*,‡,§

NANO LETTERS

2005
Vol. 5, No. 12
2586−2592

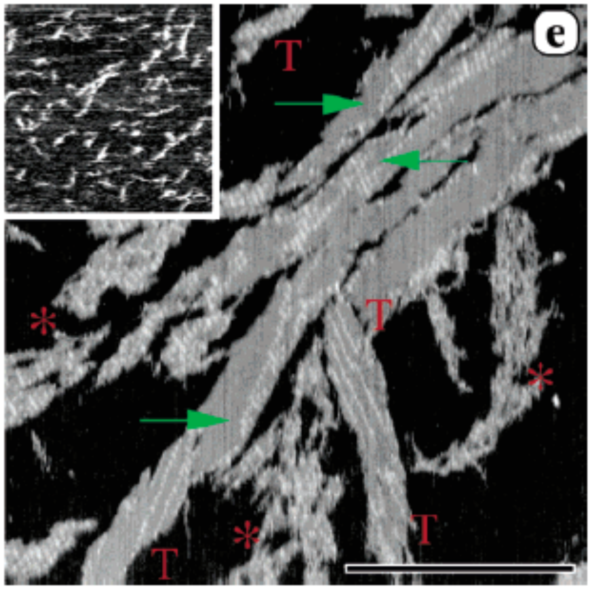
Copying and counting are useful primitive operations for computation and construction. We have made DNA crystals that copy and crystals that count as they grow. For counting, 16 oligonucleotides assemble into four DNA Wang tiles that subsequently crystallize on a polymeric nucleating scaffold strand, arranging themselves in a binary counting pattern that could serve as a template for a molecular electronic demultiplexing circuit. Although the yield of counting crystals is low, and per-tile error rates in such crystals is roughly 10%, this work demonstrates the potential of algorithmic self-assembly to create complex nanoscale patterns of technological interest. A subset of the tiles for counting form information-bearing DNA tubes that copy bit strings from layer to layer along their length.

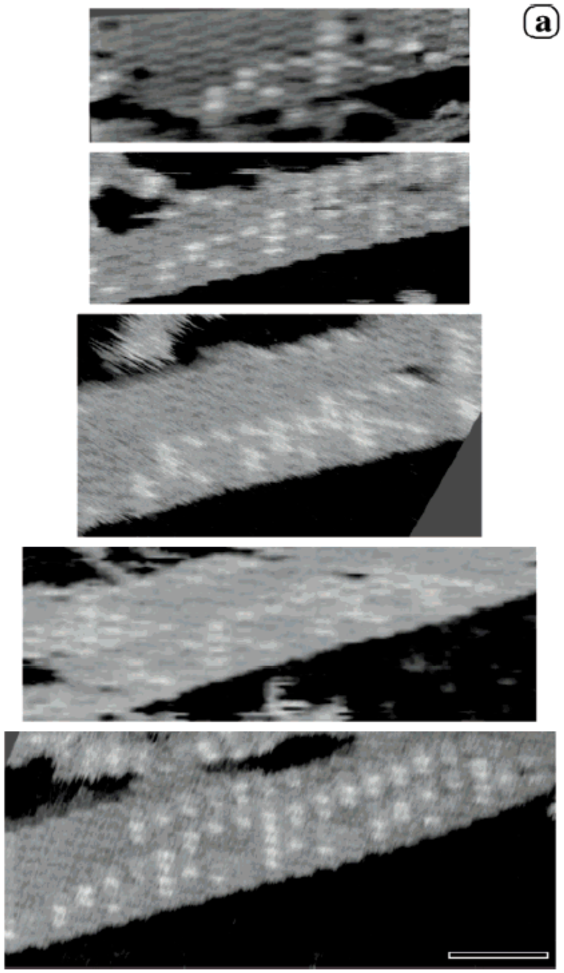


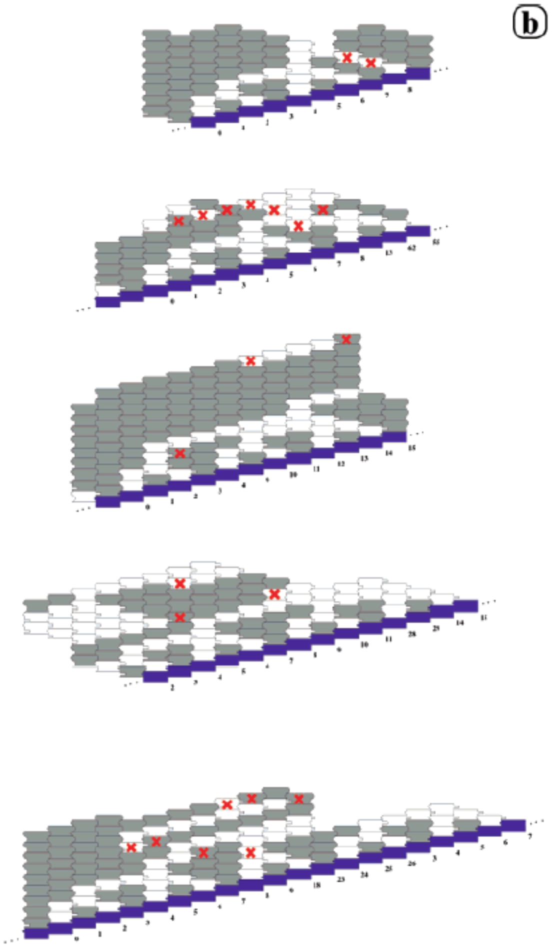
Caltech Counting Lattice

NANO
LETTERS

2005
Vol. 5, No. 12
2586−2592

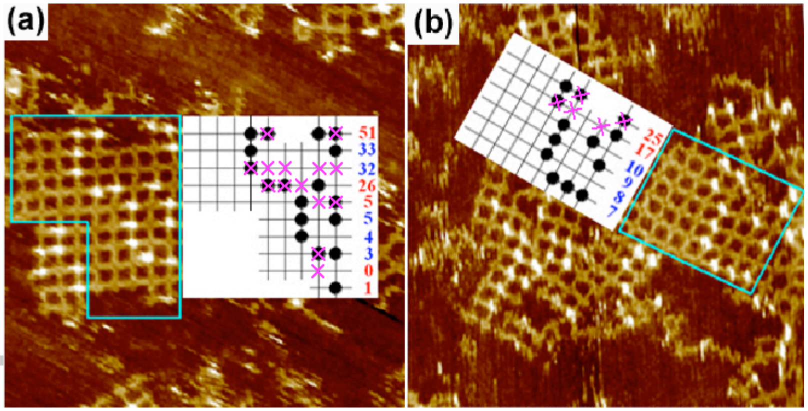


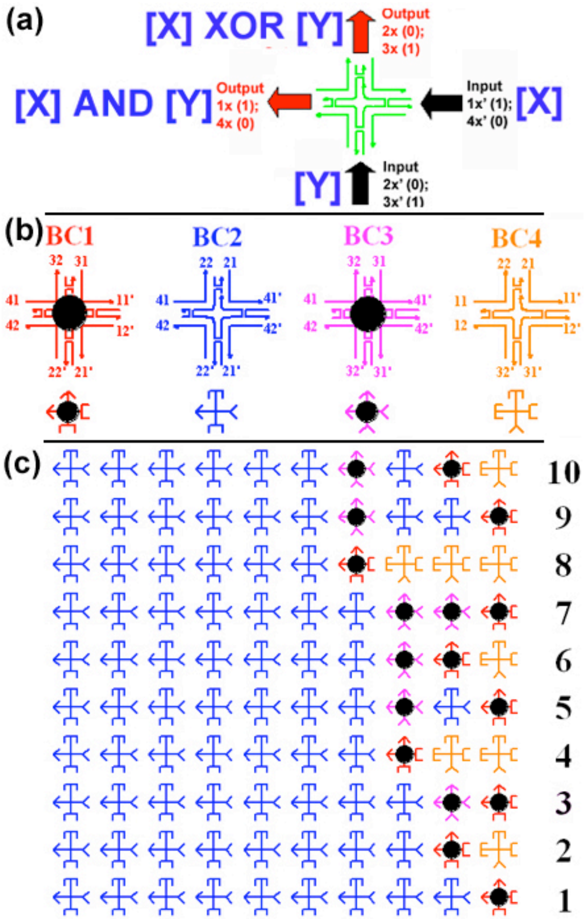




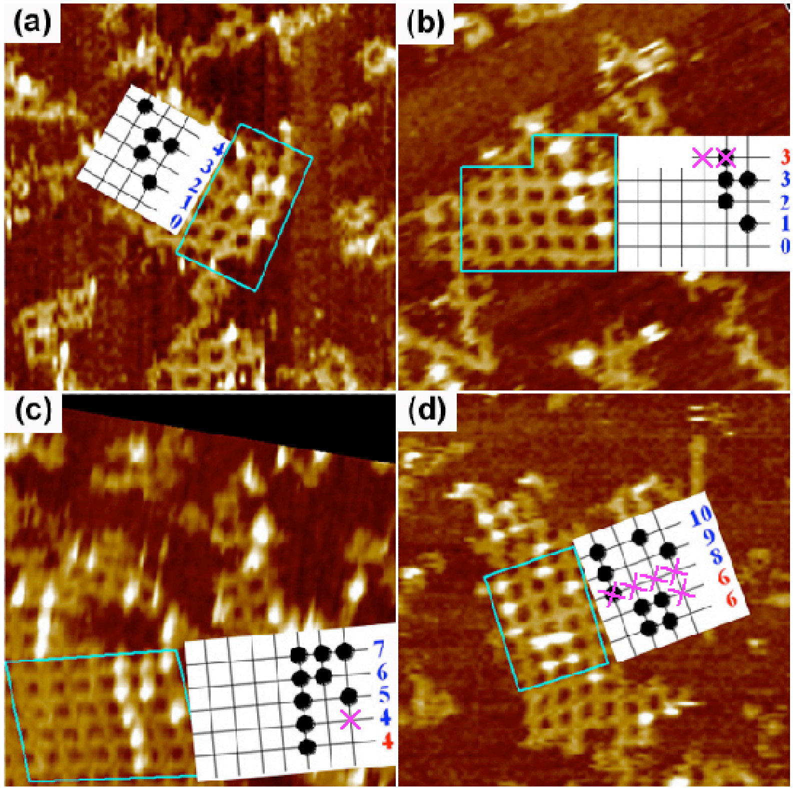
Duke Counting Lattice

Without ratio control:





With ratio control:



DNA-based Computing

- Adleman - Hamiltonian Path (7 node)
- Biochemical manipulation
- Sticker model, RNA, Whiplash PCR
- Surface-based computation
- Algorithmic self-assembly
- Autonomous systems

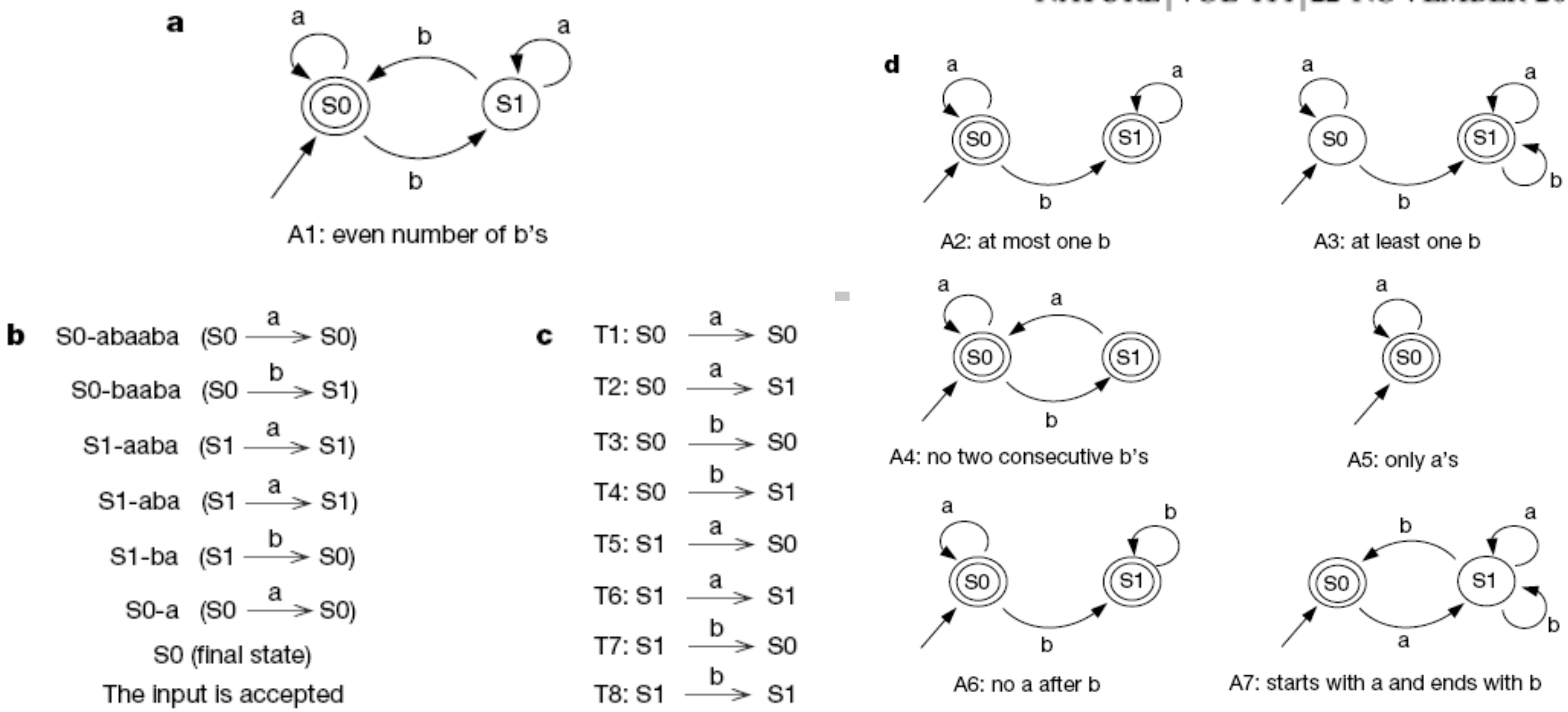
Programmable and autonomous computing machine made of biomolecules

NATURE | VOL 414 | 22 NOVEMBER 2001

Yaakov Benenson[*†], Tamar Paz-Elizur[†], Rivka Adar[†], Ehud Keinan[‡§], Zvi Livneh[†] & Ehud Shapiro[*†]

- Automata: devices which convert information in one form into another according to a definite procedure.
- Finite automaton: special case of a Turing machine which scans a tape for data and switches state according to some program.
- This DNA computer does not require manipulation by a technician or even thermal cycling.
- Hardware: restriction endonuclease, ligase.
- Software: dsDNA.
- Input data: dsDNA.

NATURE | VOL 414 | 22 NOVEMBER 2001



S0-abaaba (S0 $\xrightarrow{a}$ S0)

S0-baaba (S0 $\xrightarrow{b}$ S1)

S1-aaba (S1 $\xrightarrow{a}$ S1)

S1-aba (S1 $\xrightarrow{a}$ S1)

S1-ba (S1 $\xrightarrow{b}$ S0)

S0-a (S0 $\xrightarrow{a}$ S0)

S0 (final state)

The input is accepted

T1: S0 $\xrightarrow{a}$ S0

T2: S0 $\xrightarrow{a}$ S1

T3: S0 $\xrightarrow{b}$ S0

T4: S0 $\xrightarrow{b}$ S1

T5: S1 $\xrightarrow{a}$ S0

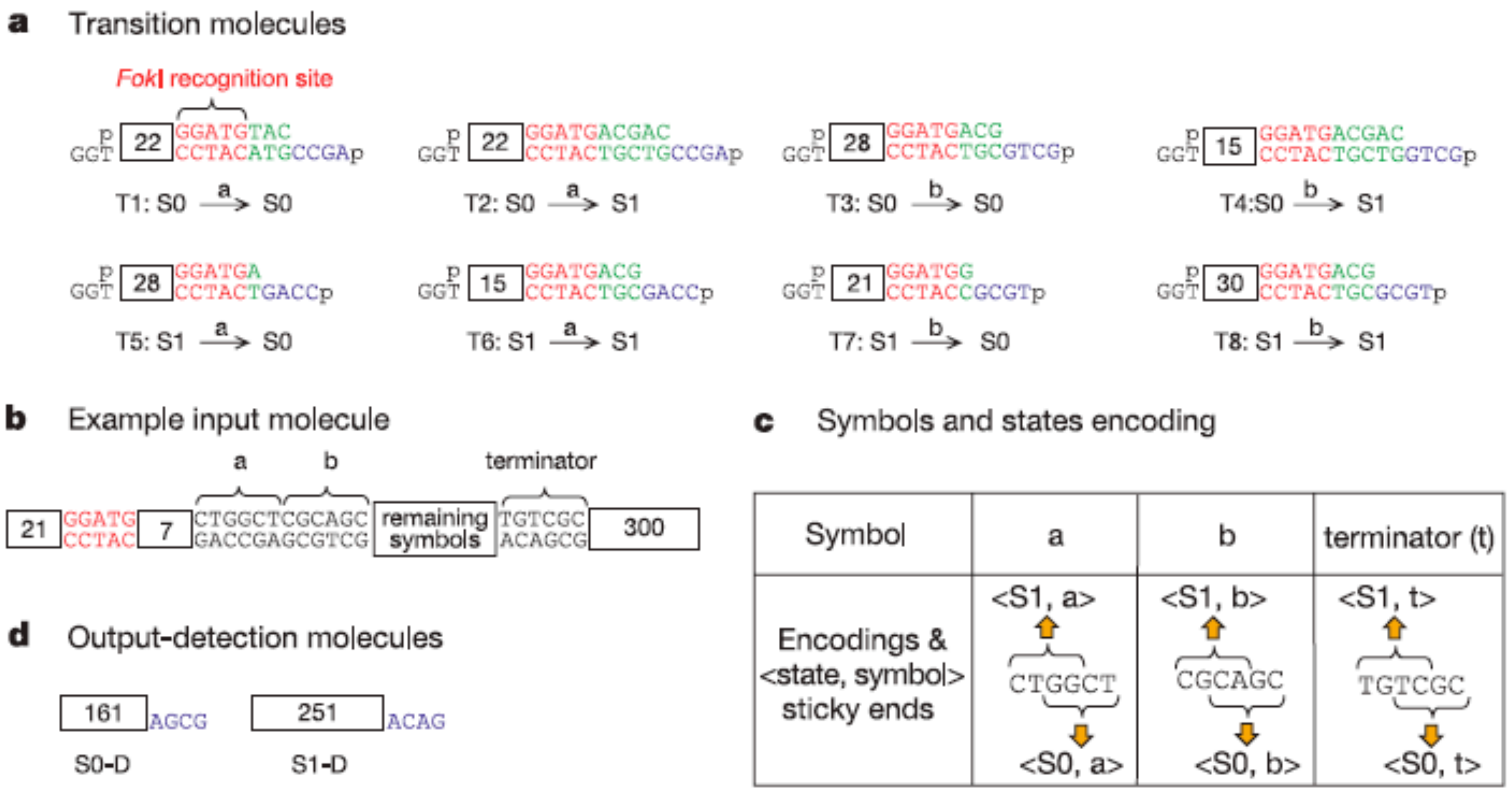
T6: S1 $\xrightarrow{a}$ S1

T7: S1 $\xrightarrow{b}$ S0

T8: S1 $\xrightarrow{b}$ S1

Figure 1 Finite automata with two states (S0 and S1) and two symbols (a and b). **a**, Diagram representing the automaton A1 accepting inputs with an even number of b symbols. Incoming unlabelled arrow represents the initial state, labelled arrows represent transition rules, and the double circle represents an accepting state. **b**, An example computation over abaaba. Each row represents an intermediate configuration, showing the current state of the automaton and the remaining symbols to be processed. The transition rule taking a configuration to its successor is shown on the right. **c**, A list of all eight possible transition rules of a two-state two-symbol automaton. **d**, Six other automata programs used to test the molecular implementation and the sets of inputs they accept. Non-deterministic automaton A7 has two transitions T7 and T8 applicable to the same configuration. A computation of A7 ending in an accepting state uses T8 for all b symbols except the last, and uses T7 for the last b.

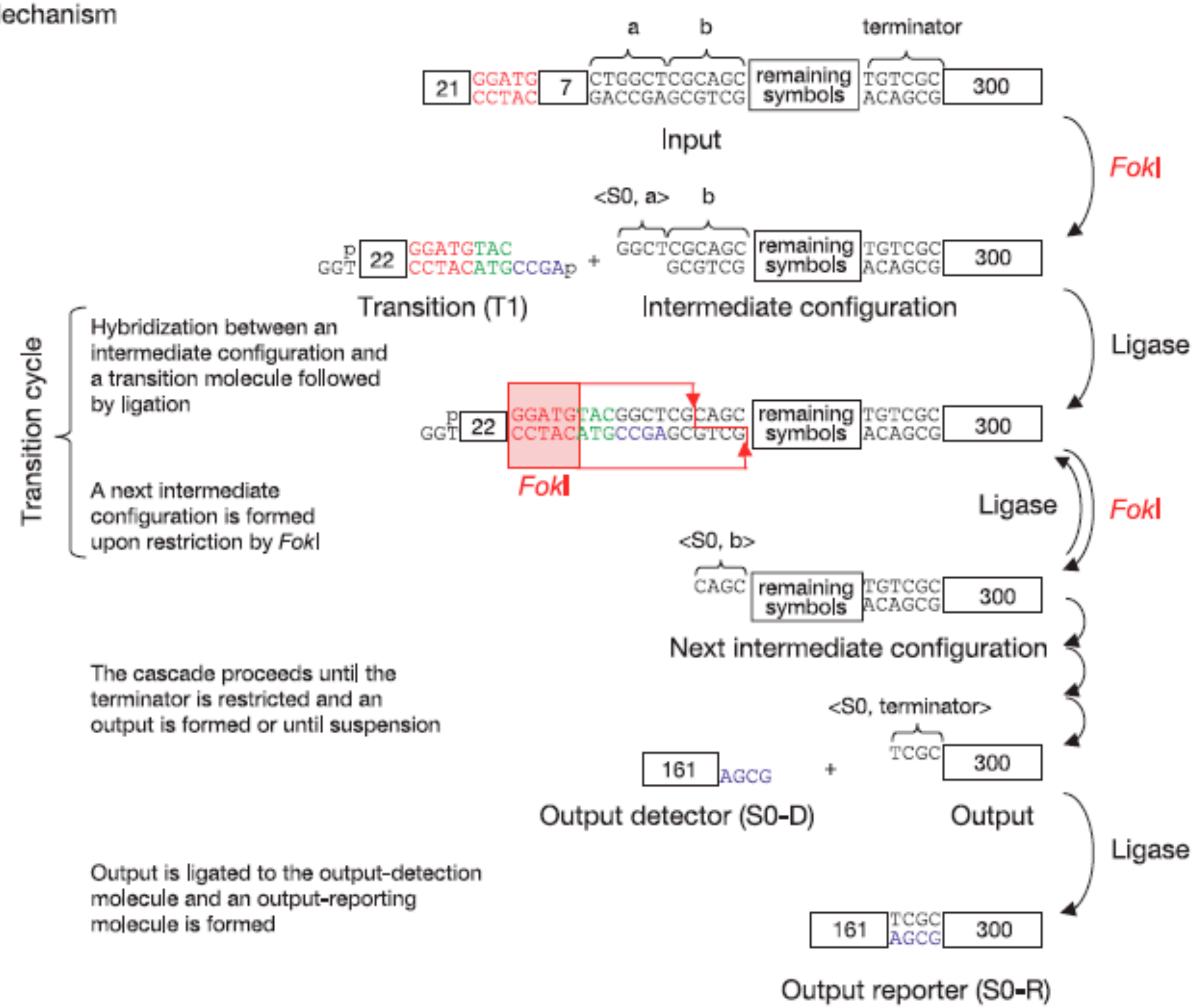
NATURE | VOL 414 | 22 NOVEMBER 2001


a Transition molecules
FokI recognition site
p GGT 22 GGATGTAC CCTACATGCCGAp
T1: S0 a→ S0
p GGT 22 GGATGACGAC CCTACTGCTGCCGAp
T2: S0 a→ S1
p GGT 28 GGATGACG CCTACTGCGTCGp
T3: S0 b→ S0
p GGT 15 GGATGACGAC CCTACTGCTGGTCGp
T4:S0 b→ S1
p GGT 28 GGATGA CCTACTGACCp
T5: S1 a→ S0
p GGT 15 GGATGACG CCTACTGCGACCp
T6: S1 a→ S1
p GGT 21 GGATGG CCTACCGCGTp
T7: S1 b→ S0
p GGT 30 GGATGACG CCTACTGCGCGTp
T8: S1 b→ S1
b Example input molecule
a
b
terminator
21 GGATG CCTAC 7 CTGGCTCGCAGC GACCGAGCGTCG remaining symbols TGTCGC ACAGCG 300
c Symbols and states encoding
d Output-detection molecules
161 AGCG
S0-D
251 ACAG
S1-D

Symbol	a	b	terminator (t)
Encodings & <state, symbol> sticky ends	<S1, a> CTGGCT <S0, a>	<S1, b> CGCAGC <S0, b>	<S1, t> TGTCGC <S0, t>

NATURE | VOL 414 | 22 NOVEMBER 2001

e Mechanism



Hybridization between an intermediate configuration and a transition molecule followed by ligation

A next intermediate configuration is formed upon restriction by *Fok*I

The cascade proceeds until the terminator is restricted and an output is formed or until suspension

Output is ligated to the output-detection molecule and an output-reporting molecule is formed

Movie 1

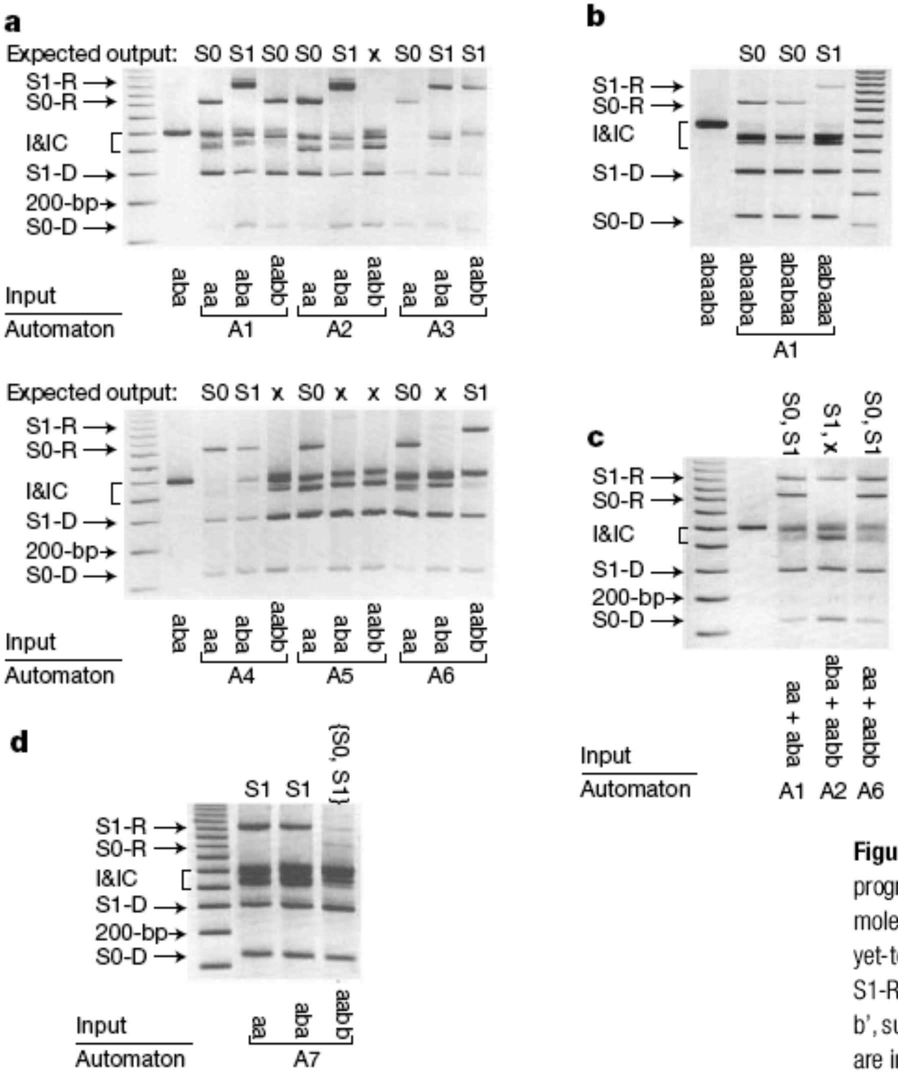


Figure 3 Running automaton programs on inputs. **a**, Experimental testing of automaton programs A1–A6 (Fig. 1a and d). The molecules in each lane are the output-detection molecules S0-D and S1-D, molecules encoding intermediate configurations capped by yet-to-be-processed input molecules and followed by output-reporting molecules S0-R or S1-R, which are missing in case the computation suspends (for example, A2, 'at most one b', suspends on aabb). Expected locations of different species and a size marker of 200 bp are indicated by arrows to the left. Input molecules are specified below each lane, and expected outputs (S0, S1 or suspension (x)) are indicated above each lane. Reactions with the same software are grouped in triplets, the software indicated below. I&IC, input and intermediate configurations. **b**, Computations with A1 software performed over 6-symbol-long input molecules. All inputs are of the same size, shown on a reference lane. **c**, Parallel computation. Input mixture composition and the software used are indicated below the lanes. **d**, Demonstration of a non-deterministic computation.

NATURE | VOL 414 | 22 NOVEMBER 2001

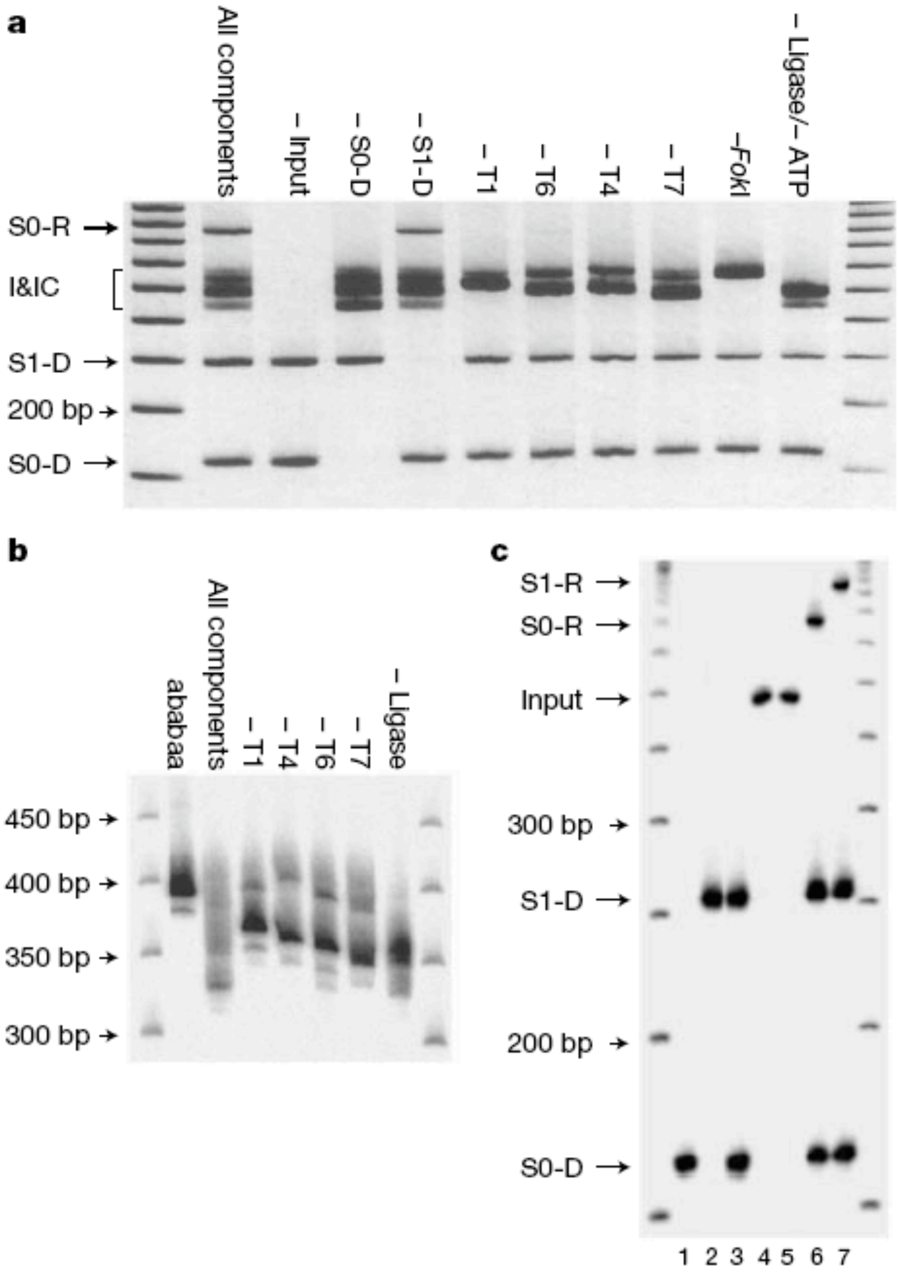


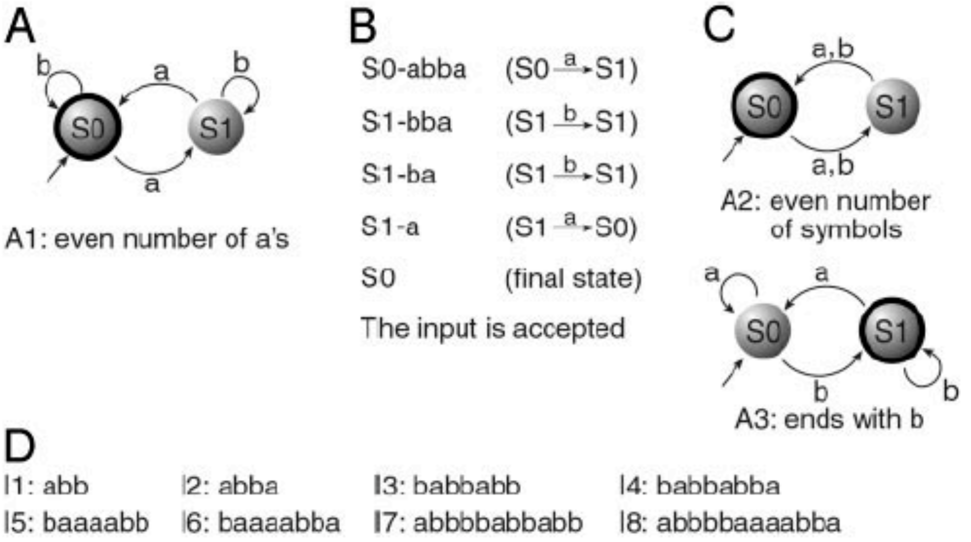
Figure 4 Verification of the operation mechanism. **a**, Identification of the essential components. Each lane is a computation reaction with one component omitted (above). The ligase or ATP-free reactions were identical and are represented by one lane. **b**, Close inspection of the reaction intermediates analysed by native PAGE (5%). Sample composition is indicated above the gel image. The ababaa lane contains labelled input. 'All components' lane is an output-detectors-free A1 computation, performed for reference. All other lanes lack the indicated components. The size markers are indicated to the left. **c**, An estimation of system fidelity analysed by native PAGE (6%). The composition of different lanes is as follows: 1, pure labelled S0-D; 2, pure labelled S1-D; 3, input-free computation reaction; 4, pure labelled ababaa; 5, pure labelled aabaaa; 6, a computation reaction over ababaa with unlabelled input and labelled output-detection molecules; 7, a similar reaction over aabaaa. Locations of the different components and the size markers of 200 bp and 300 bp are indicated on the left.

DNA molecule provides a computing machine with both data and fuel

PNAS | March 4, 2003 | vol. 100 | no. 5 | 2191–2196

Yaakov Benenson*[†‡], Rivka Adar[†‡], Tamar Paz-Elizur[†], Zvi Livneh[†], and Ehud Shapiro*[†§]

Departments of *Computer Science and Applied Mathematics and [†]Biological Chemistry, Weizmann Institute of Science, Rehovot 76100, Israel



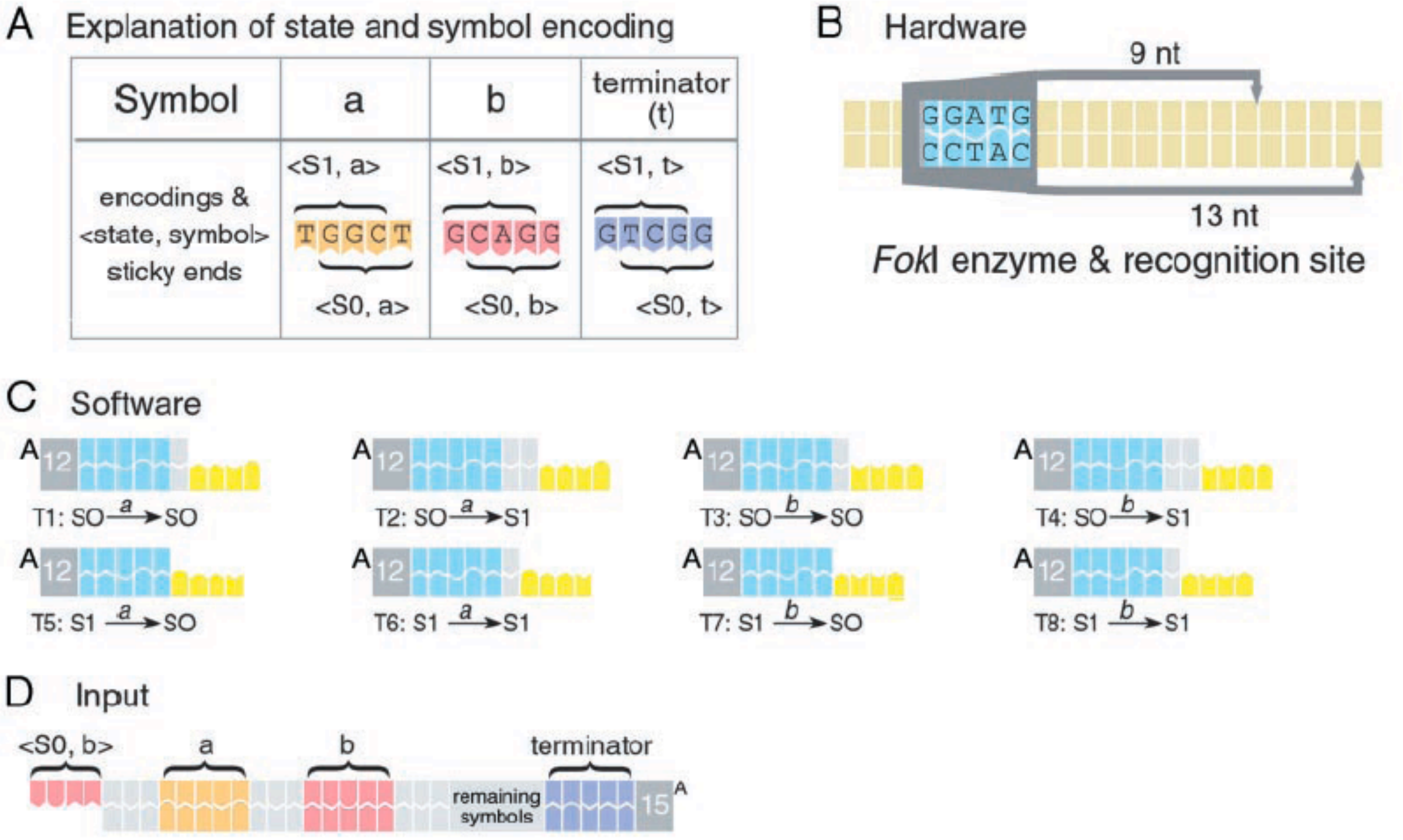
S0-abba	(S0 $\xrightarrow{a}$ S1)
S1-bba	(S1 $\xrightarrow{b}$ S1)
S1-ba	(S1 $\xrightarrow{b}$ S1)
S1-a	(S1 $\xrightarrow{a}$ S0)
S0	(final state)

The input is accepted

I1: abb	I2: abba	I3: babbabb	I4: babbabba
I5: baaaabb	I6: baaaabba	I7: abbbbabbabb	I8: abbbbaaaabba

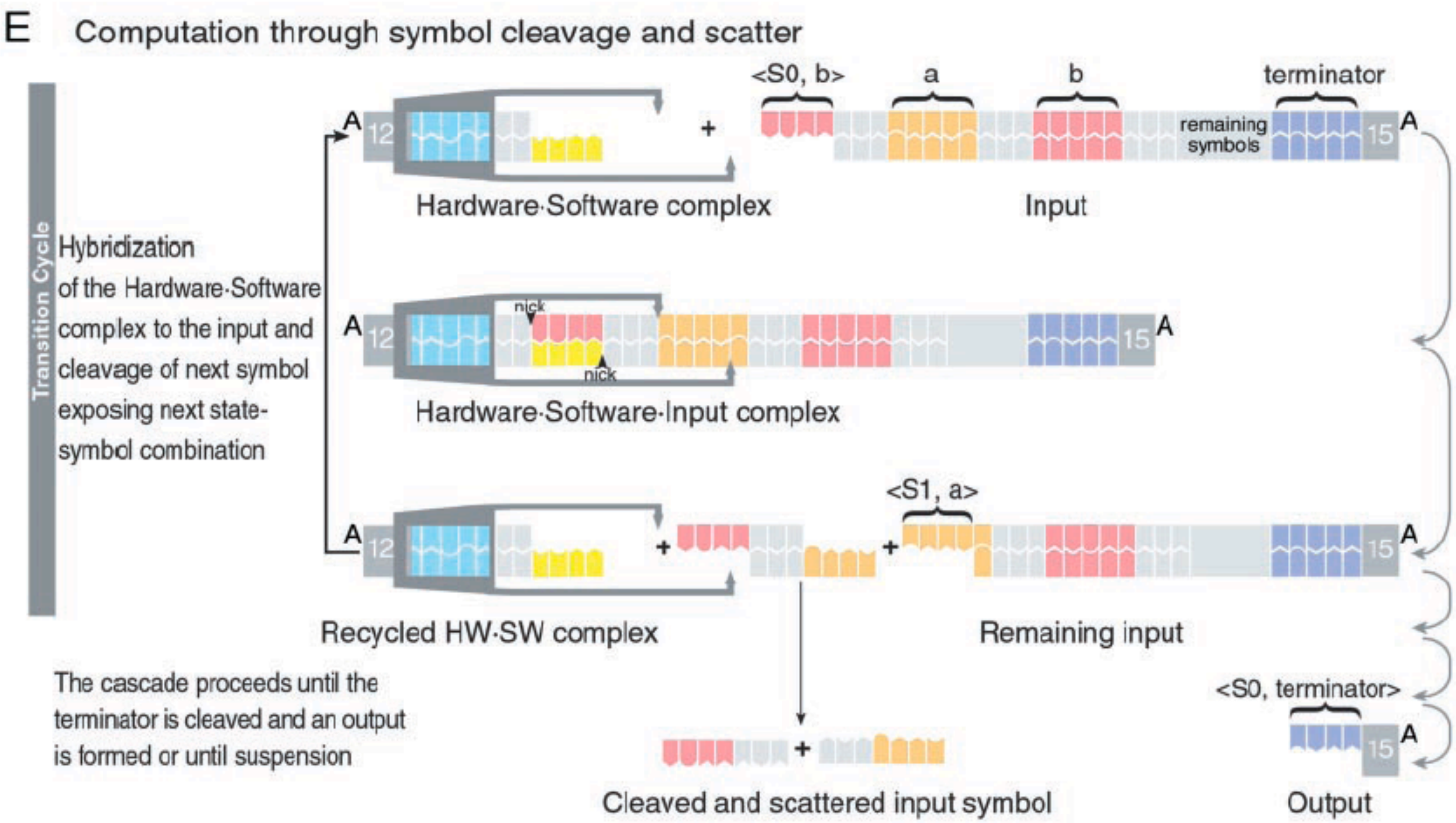
Fig. 1. Finite automata and inputs for which we show molecular realizations. (*A*) Automaton A1. Incoming unlabeled arrow marks the initial state, and labeled arrows represent transition rules. A double circle represents an accepting state. (*B*) A computation of A1 scanning and digesting the input abba. Each row represents an intermediate configuration showing current state and remaining input. The transition rule applied is shown in brackets. (*C*) The two other automata for which we demonstrate molecular operation. (*D*) The set of inputs used as fuel.

PNAS | March 4, 2003 | vol. 100 | no. 5 | 2191–2196



Symbol	a	b	terminator (t)
encodings & <state, symbol> sticky ends	<S1, a> TGGCT <S0, a>	<S1, b> GCAGG <S0, b>	<S1, t> GTCGG <S0, t>

PNAS | March 4, 2003 | vol. 100 | no. 5 | 2191–2196



Hybridization of the Hardware·Software complex to the input and cleavage of next symbol exposing next state-symbol combination

The cascade proceeds until the terminator is cleaved and an output is formed or until suspension

Fig. 2. A molecular finite automaton that uses input as fuel. (*A*) Encoding of a, b, and terminator (sense strands) and the <state, symbol> interpretation of exposed 4-nt sticky ends, the leftmost representing the current symbol and the state S1, similarly the rightmost for S0. (*B*) Hardware: The *Fok*I restriction enzyme, which recognizes the sequence GGATG and cleaves 9 and 13 nt apart on the 5′ → 3′ and 3′ → 5′ strands, respectively. (*C*) Software: Each DNA molecule realizes a different transition rule by detecting a current state and symbol and determining a next state. It consists of a <state, symbol> detector (yellow), a *Fok*I recognition site (blue), and a spacer (gray) of variable length that determines the *Fok*I cleavage site inside the next symbol, which in turn defines the next state. Empty spacers effect S1 to S0 transition, 1-bp spacers maintain the current state, and 2-bp spacers transfer S0 to S1. (*D*) Input: The exposed sticky end at the 5′ terminus of the DNA molecule encodes the initial state and first symbol. Each symbol is encoded with 5 bp separated by 3-bp spacers. (*E*) Suggested mechanism of operation of the automaton. The computation proceeds via a cascade of transition cycles, each cleaving and scattering one input symbol, exemplified with the input molecule bab in the initial state S0 and the transition $S0 \xrightarrow{b} S1$. Both hardware and software molecules are recycled.

Movie 2

PNAS | March 4, 2003 | vol. 100 | no. 5 | 2191–2196

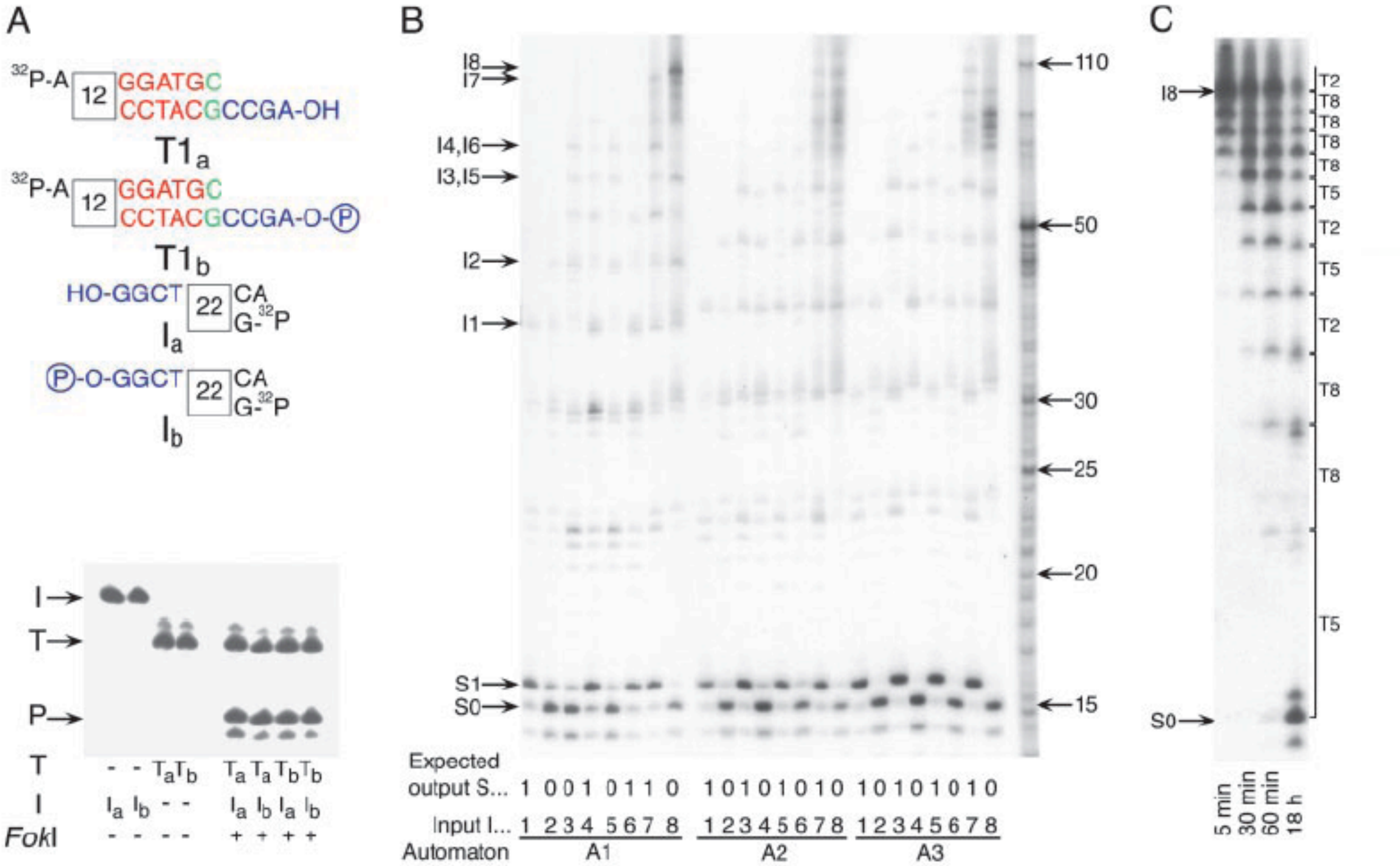


Fig. 3. Experimental results and mechanism analysis of the molecular automaton. (*A*) A demonstration that a computation does not require ligase. Different variants of the software molecule T1 (T1$_{a}$, nonphosphorylated, and T1$_{b}$, phosphorylated) and the input (I$_{a}$, nonphosphorylated and I$_{b}$, phosphorylated) were incubated with the hardware (*Fok*I) at 8°C for 10 min. Input, software, and hardware concentrations were all 1 μM. Reaction components are shown below the lanes, and the locations of software molecule (T), input (I), and product (P) are indicated by arrows. (*B*) Executing automata A1–A3 (Fig. 1 *A* and *C*) on inputs I1–I8 (Fig. 1*D*). Input, software, and hardware concentrations were 1, 4, and 4 μM, respectively. Reactions were set in 10 μl and incubated at 8°C for 20 min. Each lane contains inputs, intermediate configurations, and output bands at the indicated locations. The programs, inputs, and expected outputs are indicated below each lane. Location of unprocessed input is shown on the left. Size markers are indicated by arrows on the right. (*C*) Software reusability with the four-transition automaton A1 applied to input I8 with each software molecule taken at 0.075 molar ratio to the input. Input, software, and hardware concentrations were 1, 0.3 (0.075 μM each kind), and 1 μM, respectively. After 18 h, molecules T2, T5, and T8 performed on the average 29, 21, and 54 transitions each. Input and output locations are indicated on the left, and intermediates and software molecules applied at each step are on the right.

Deoxyribozyme-Based Logic Gates

Milan N. Stojanovic,[*,†] Tiffany Elizabeth Mitchell,[†] and Darko Stefanovic[‡,§]

Contribution from the Division of Clinical Pharmacology and Experimental Therapeutics, Department of Medicine, Columbia University, New York, and Department of Computer Science, University of New Mexico, Albuquerque, New Mexico

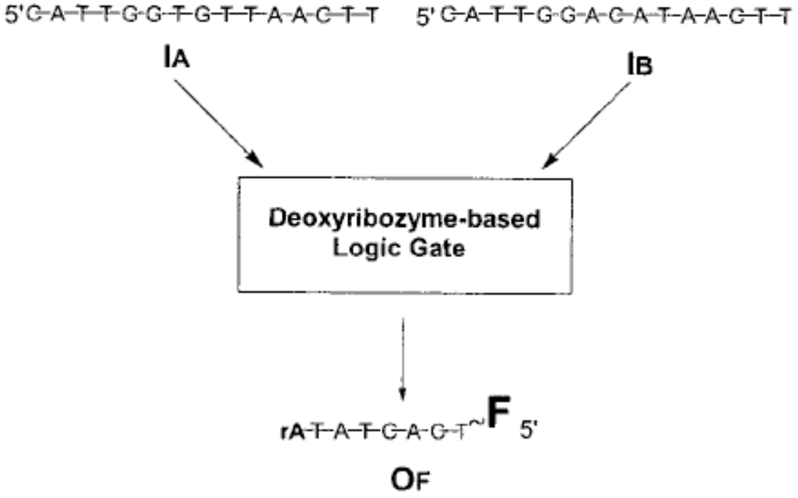

5'C-A-T-T-G-G-T-G-T-T-A-A-C-T-T
5' C-A-T-T-G-G-A-C-A-T-A-A-C-T-T
IA
IB
Deoxyribozyme-based Logic Gate
rA-T-A-T-C-A-G-T~F 5'
OF

Figure 1. Basic concept: input oligonucleotides I_A and I_B result in the presence or absence of output fluorescent product O_F, depending on the interactions with deoxyribozyme-based logic gates

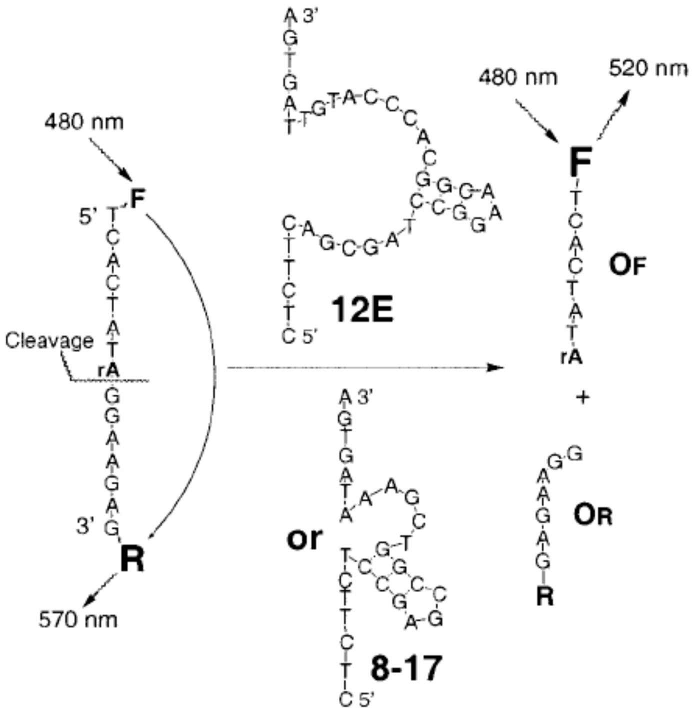
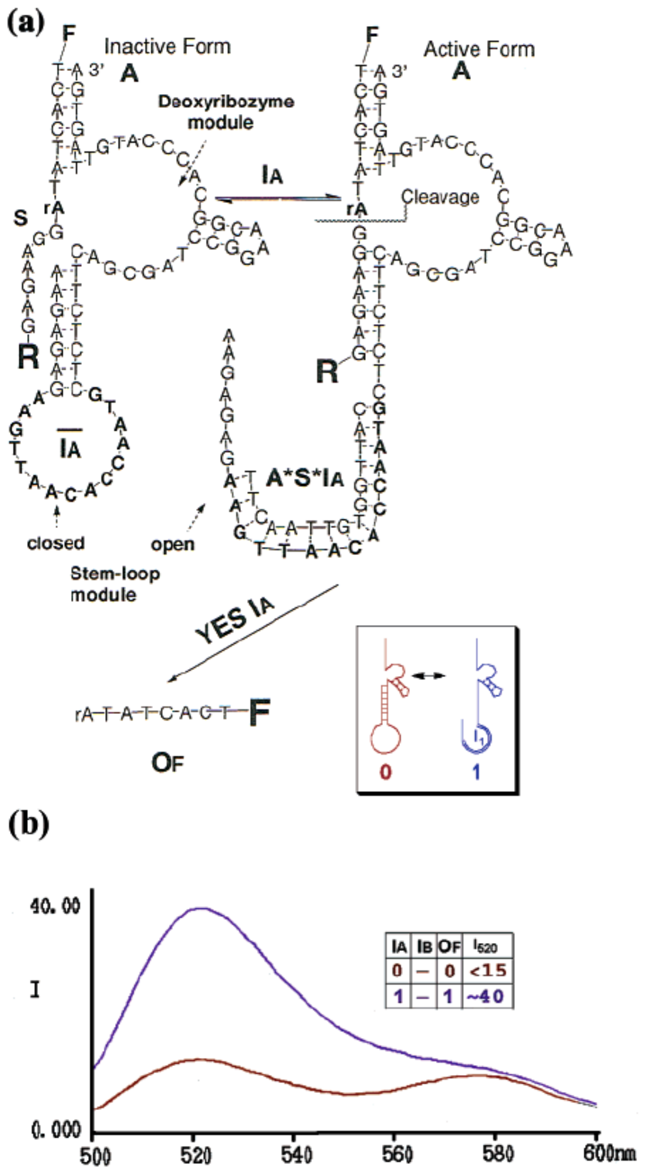


Figure 2. Fluorogenic cleavage of double end-labeled substrate by deoxyribozymes **E6** or **8-17** into products O_F and O_R. Fluorescein (**F**) emission is quenched by distance-dependent fluorescence resonance energy transfer to tetramethylrhodamine (**R**), and upon cleavage fluorescence increases (larger font **F**).

J. AM. CHEM. SOC. 2002, *124*, 3555 ■ 3555

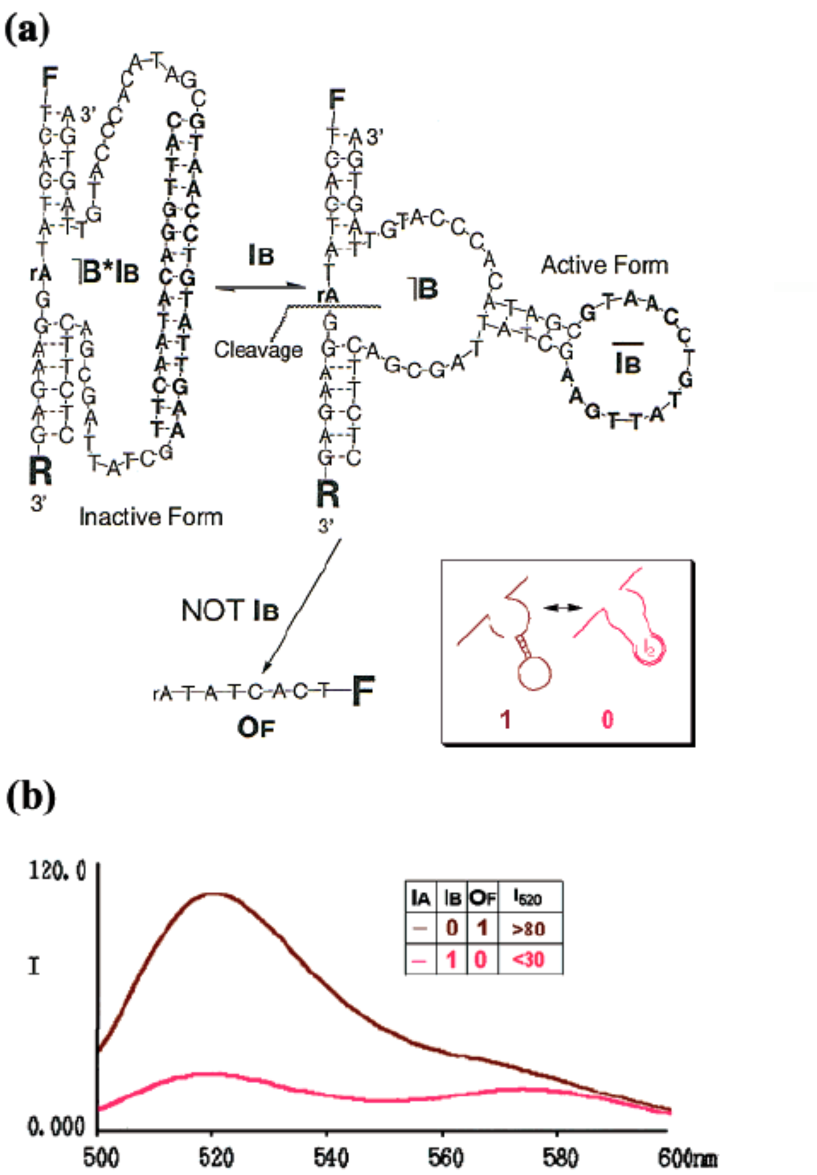
DNA logic gates

J. AM. CHEM. SOC. 2002, *124*, 3555 ■ 3555



IA	IB	OF	I520
0	–	0	<15
1	–	1	~40

Figure 3. (a) Single input sensor gate (A) is activated by the input

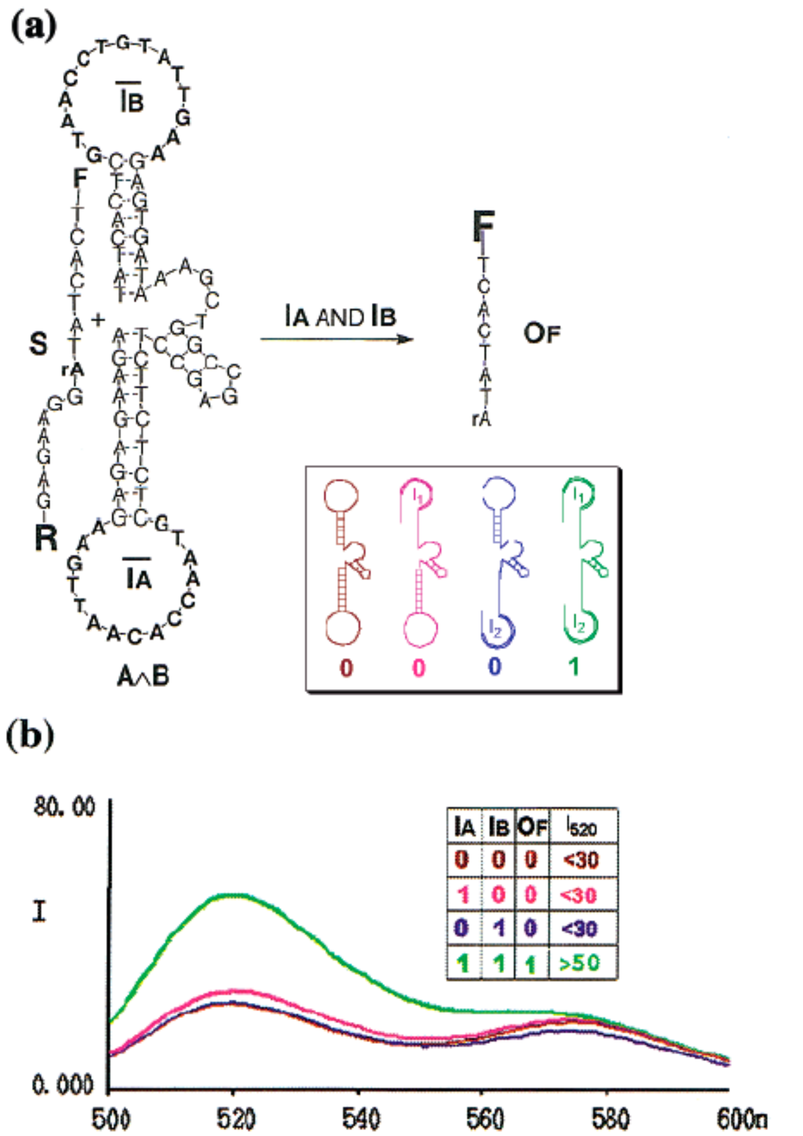


IA	IB	OF	I520
–	0	1	>80
–	1	0	<30

Figure 4. (a) Single-input NOT gate (¬B) is constructed through

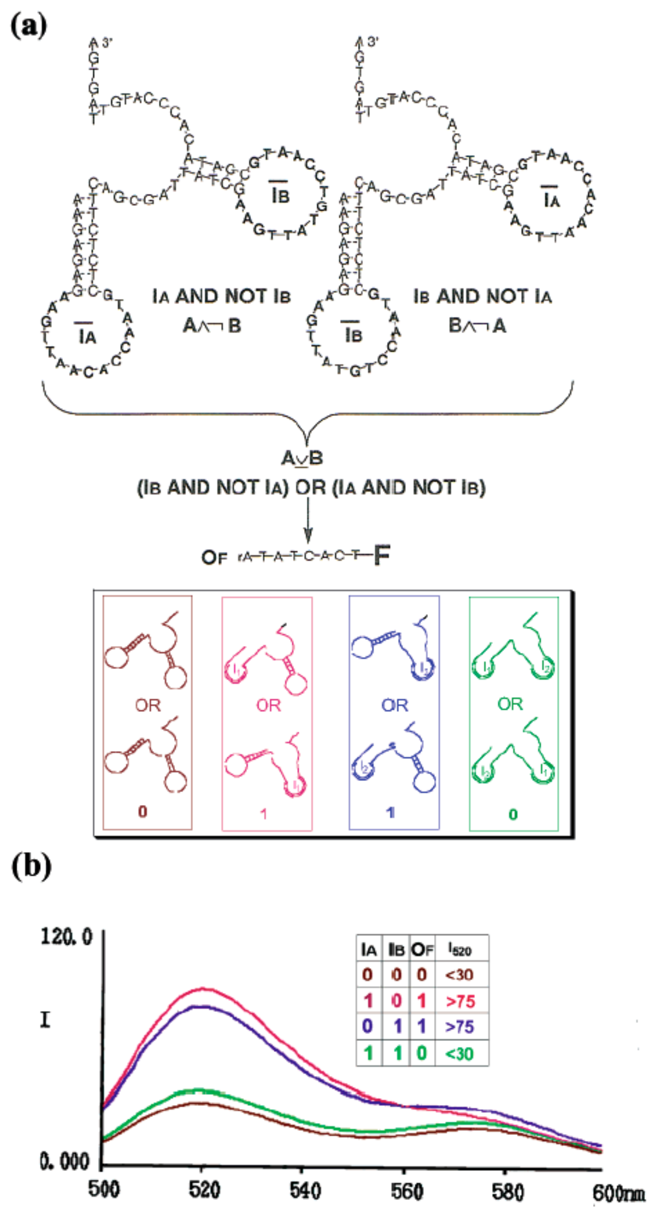
DNA logic gates

J. AM. CHEM. SOC. 2002, *124*, 3555 ■ 3555



IA	IB	OF	I520
0	0	0	<30
1	0	0	<30
0	1	0	<30
1	1	1	>50

Figure 5. (a) AND gate (A∧B) is constructed through attachment of two

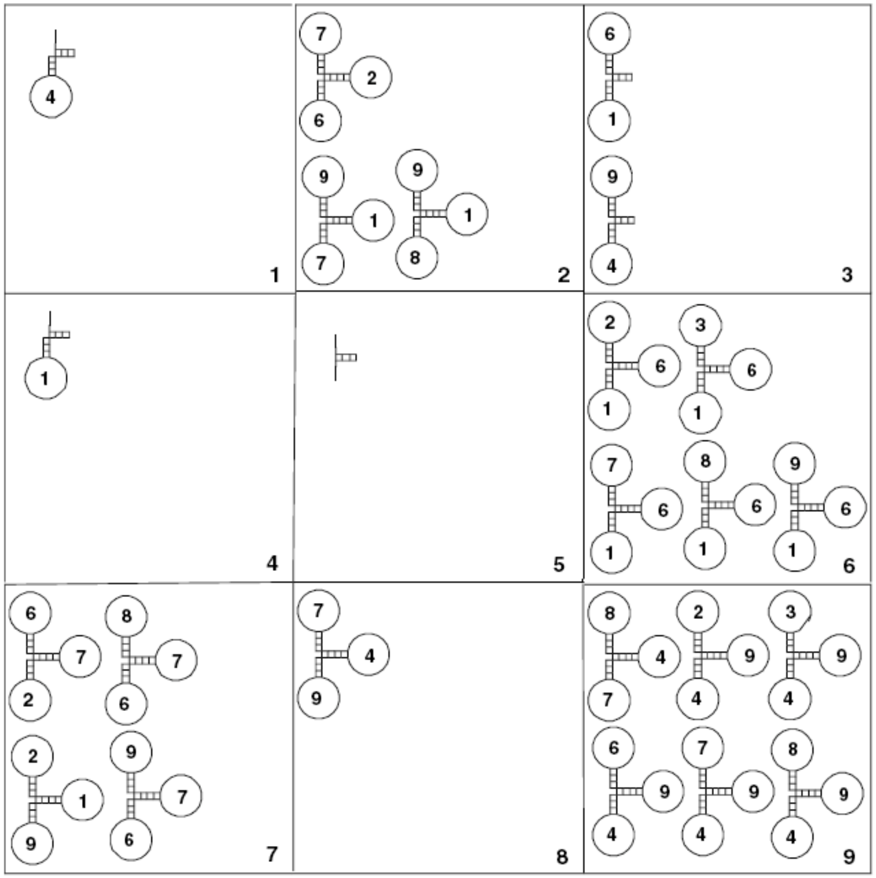


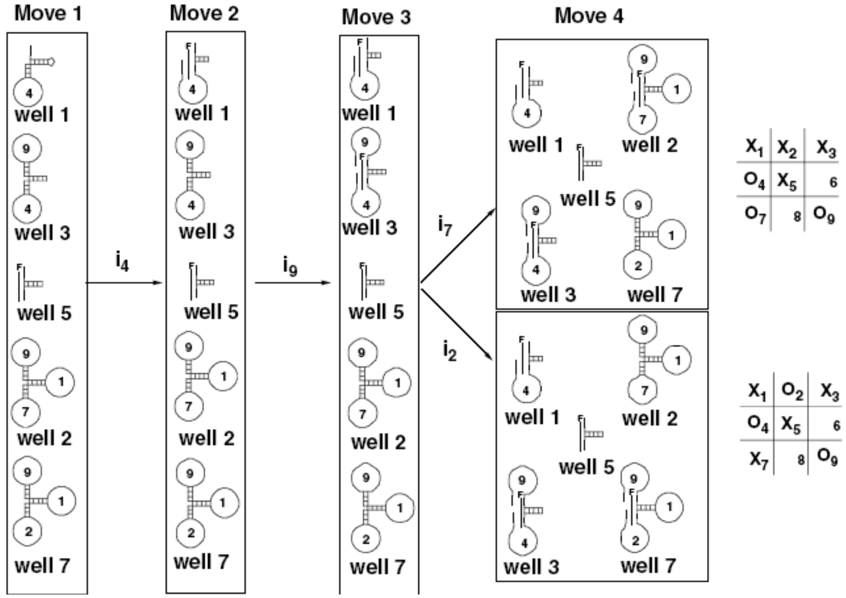
IA	IB	OF	I520
0	0	0	<30
1	0	1	>75
0	1	1	>75
1	1	0	<30

Figure 7. (a) XOR gate (A⊻B) as a combination of A∧¬B and B∧¬A

Deoxyribozyme-based tictactoe

Stojanovic & Stefanovic, 2003 *Nat. Biotechnol.* **21**, 1069





X_1	X_2	X_3
O_4	X_5	6
O_7	8	O_9

X_1	O_2	X_3
O_4	X_5	6
X_7	8	O_9

Fig. 14.15 Two representative games (differing in the third move by the human and the fourth move by the automaton) representing the winning strategy implemented by automaton. We show only those gates that participate in these two games; all other gates remain silent. In the first move by the automaton, a constitutively active gate in well 5 gives a fluorescent signal. After the human's first move, conveyed to the automaton by the addition of i_4, the gate in well 1 is activated. The second human move (i_8) triggers the fluorogenic cleavage in well 3. The human has no choice now, as the automaton has formed a fork: any move by the human will trigger a three-in-a-row. We show two human moves, an attempt to block the diagonal three-in-a-row by adding i_7 to all wells, which leads to a vertical three-in-a-row by activating i_7ANDi$_9$ANDNOTi$_1$, and addition of i_2, which leads to a diagonal three-in-a-row by activating i_2ANDi$_9$ANDNOTi$_1$.