Autonomous Enzymic Finite State Molecular Computers

Work by Yaakov Benenson & Ehud Shapiro

Department of Computer Science and Applied Math &
Department of Biological Chemistry
Weizmann Institute of Science, Rehovot, Israel

• Programmable and autonomous computing machine made of Biomolecules, Nature, 2001

• DNA molecule provides a computing machine with both data and fuel, PNAS, 2003

• Stochastic computing with biomolecular automata, PNAS, 2004

• An autonomous molecular computer for logical control of gene expression, Nature, 2004
Programmable and autonomous computing machine made of Biomolecules,

Nature, 2001

Automaton $A1$ accepting inputs with an even number of $b$'s

$A1$: even number of $b$'s

[Benenson & Shapiro, Nature, 2001]
Turing Machine and Finite Automaton

State transition rules:
- $S_0, 0 \rightarrow S_0$
- $S_1, 0 \rightarrow S_1$
- $S_0, 1 \rightarrow S_1$
- $S_1, 1 \rightarrow S_0$

State transition diagram:

[Benenson & Shapiro, Nature, 2001]
A1: even number of b's

S0-abaaba  (S0 → S0)
S0-baaba   (S0 → S1)
S1-aaba    (S1 → S1)
S1-aba     (S1 → S1)
S1-ba      (S1 → S0)
S0-a       (S0 → S0)
S0 (final state)
The input is accepted

A2: at most one b

A3: at least one b

A4: no two consecutive b's

A5: only a's

A6: no a after b

A7: starts with a and ends with b

[Benenson & Shapiro, Nature, 2001]
Example Computation

State transition rules:

- $S_0, 0 \rightarrow S_0$
- $S_1, 0 \rightarrow S_1$
- $S_0, 1 \rightarrow S_1$
- $S_1, 1 \rightarrow S_0$

State transition diagram:

Initial state: $S_0$

State transition rules:

1. $S_0 
   \begin{array}{c}
   0 & 1 & 0 & 0 & 1 & 0 & 0
   \end{array}
   \rightarrow
   \begin{array}{c}
   S_0
   \end{array}
   \begin{array}{c}
   \rightarrow
   \end{array}
   \begin{array}{c}
   S_0
   \end{array}$

2. $S_0 
   \begin{array}{c}
   0 & 1 & 0 & 0 & 1 & 0 & 0
   \end{array}
   \rightarrow
   \begin{array}{c}
   S_0, 0 \rightarrow S_0
   \end{array}
   \begin{array}{c}
   \rightarrow
   \end{array}
   \begin{array}{c}
   S_0
   \end{array}$

3. $S_0 
   \begin{array}{c}
   0 & 1 & 0 & 0 & 1 & 0 & 0
   \end{array}
   \rightarrow
   \begin{array}{c}
   S_0, 1 \rightarrow S_1
   \end{array}
   \begin{array}{c}
   \rightarrow
   \end{array}
   \begin{array}{c}
   S_1
   \end{array}$

4. $S_1 
   \begin{array}{c}
   0 & 1 & 0 & 0 & 1 & 0 & 0
   \end{array}
   \rightarrow
   \begin{array}{c}
   S_1, 0 \rightarrow S_1
   \end{array}
   \begin{array}{c}
   \rightarrow
   \end{array}
   \begin{array}{c}
   S_1
   \end{array}$

5. $S_1 
   \begin{array}{c}
   0 & 1 & 0 & 0 & 1 & 0 & 0
   \end{array}
   \rightarrow
   \begin{array}{c}
   S_1, 1 \rightarrow S_0
   \end{array}
   \begin{array}{c}
   \rightarrow
   \end{array}
   \begin{array}{c}
   S_0
   \end{array}$

6. $S_0 
   \begin{array}{c}
   0 & 1 & 0 & 0 & 1 & 0 & 0
   \end{array}
   \rightarrow
   \begin{array}{c}
   S_0, 0 \rightarrow S_0
   \end{array}
   \begin{array}{c}
   \rightarrow
   \end{array}
   \begin{array}{c}
   S_0
   \end{array}$

Final state: $S_0$

$<S_0, 0>$

[Benenson&Shapiro, Nature, 2001]
An example computation over *abaaba*

- **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*

A list of all 8 possible transition rules

T1: S0 \( \rightarrow \) a \( \rightarrow \) S0

T2: S0 \( \rightarrow \) a \( \rightarrow \) S1

T3: S0 \( \rightarrow \) b \( \rightarrow \) S0

T4: S0 \( \rightarrow \) b \( \rightarrow \) S1

T5: S1 \( \rightarrow \) a \( \rightarrow \) S0

T6: S1 \( \rightarrow \) a \( \rightarrow \) S1

T7: S1 \( \rightarrow \) b \( \rightarrow \) S0

T8: S1 \( \rightarrow \) b \( \rightarrow \) S1

[Benenson&Shapiro, Nature, 2001]
Automata programs used to test the molecular implementation

A2: at most one

A3: at least one b

A4: no two consecutive b’s

A5: only a’s

A6: no a after b

A7: starts with a and ends with b

[Benenson & Shapiro, Nature, 2001]
Molecular realization of Finite Automata

• Input: DNA

• Software: DNA

• Hardware: Class-II restriction enzyme FokI, DNA Ligase, ATP as fuel

[Benenson & Shapiro, Nature, 2001]
Molecular Scale Computation

The automaton is so small that $10^{12}$ automata sharing the same software run independently and in parallel on inputs (which could in principle be distinct) in 120 ml solution at room temperature. Their combined rate is $10^9$ transitions per second, their transition fidelity is greater than 99.8%, and together they consume less than a billionth of one Watt.

[Benenson & Shapiro, Nature, 2001]
How can DNA strands to contain the symbols 0 and 1? DNA strands are usually depicted as a scroll of recurring "letters," in varied combinations, that represent DNA's constituents (four chemical bases). The team decided that the letter pattern "CTGGCT" in the input molecule would signify "0" (a in the diagram below) and "CGCAGC" would signify "1" (b in the diagram).

[Benenson & Shapiro, Nature, 2001]
The input molecule, when mixed with hardware and software molecules, also two "states". When the hardware molecule FokI, recognizing a symbol, "cuts" DNA, it leaves it with one strand longer than the other, resulting in a single-strand overhang called a "sticky end" (see diagram below). Since FokI makes its incision at the site of the symbol, the "sticky end" is what remains of the symbol. FokI may leave the symbol's "head" or "tail" attached. These are the two possible "states."

A computer that has two possible states and two possible symbols is called a two-state, two-symbol finite automaton.

Encoding: $a = \text{CTGGCT}, \quad b = \text{CGCAGC}$

$\text{CTGGCT} = 0, \quad \text{CAGC} = 1$

[Benenson & Shapiro, Nature, 2001]
Two molecules with complementary sticky ends can temporarily stick to each other (a process known as hybridization). In each processing step the input molecule hybridizes with a software molecule that has a complementary sticky end, allowing the hardware molecule Ligase to seal them together using two ATP molecules as energy (see diagram below).

Then comes Fok-I, and cleaves the input molecule again, in a location determined by the software molecule. Thus a sticky end is again exposed, encoding the next input symbol and the next state of the computation. Once the last input symbol is processed, a sticky end encoding the final state of the computation is exposed and detected, again by hybridization and ligation, by one of two "output display" molecules. The resulting molecule, which reports the output of the computation, is made visible to the human eye in a process known as gel electrophoresis.

Encoding: \(a = \text{CTGGCT, } b = \text{CGCAGC}\)

\[
\begin{align*}
\text{GGCT} &= 0, \\
\text{CAGC} &= 1
\end{align*}
\]

[Benenson&Shapiro, Nature, 2001]
Ligase and ATP use

Software is consumed

Encoding: \( a = CT\ \text{GGCT}, \quad b = CGCAGC \)

\( GGCT = 0, \quad CAGC = 1 \)
a  Transition molecules

FokI recognition site

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

b  Example input molecule

\begin{align*}
&21 \text{ GGATG} & 7 \text{ CTG} & 28 \text{ GGATGACG} & 15 \text{ CCTACTGCGTCG} \\
&7 \text{ CTTAC} & 8 \text{ CTACAG} & 15 \text{ CCTACTGCGAC} & 7 \text{ CTTAC} & 8 \text{ CTTAC} & 15 \text{ CCTACTGCGAC} & 7 \text{ CTTAC} & 8 \text{ CTTAC} & 15 \text{ CCTACTGCGAC} & 7 \text{ CTTAC} & 8 \text{ CTTAC} & 15 \text{ CCTACTGCGAC} & 7 \text{ CTTAC} & 8 \text{ CTTAC} & 15 \text{ CCTACTGCGAC} & 7 \text{ CTTAC} & 8 \text{ CTTAC} & 15 \text{ CCTACTGCGAC}
\end{align*}

\text{remaining symbols:} \text{TGTGCG} \quad \text{ACAGGC} \quad \text{300}

c  Symbols and states encoding

\begin{tabular}{|c|c|c|c|}
\hline
Symbol & \text{a} & \text{b} & \text{terminator (t)} \\
\hline
Encodings & & & \\
&\text{<S1, a>} &\text{<S1, b>} &\text{<S1, t>} \\
\text{<state, symbol> sticky ends} & \text{CTGGCT} & \text{CGCAGC} & \text{TGTGCG} \\
\hline
\end{tabular}

\begin{align*}
&\text{Output-detection molecules} \\
&\text{161 AGCG} & \text{251 ACAG} \\
&S0-D & S1-D
\end{align*}

Encoding: \text{a = CTGGCT}, \quad \text{b = CGCAGC} \\
\text{GGCT = 0, CAGC = 1} 

[Benenson&Shapiro, Nature, 2001]
---

**Mechanism**

### Transition (T1)

**Input**

- A **transition molecule** follows by **ligation**
- The cascade proceeds until the terminator is restricted and an output is formed or until suspension.

### Intermediate configuration

**Output detector (S0-D)**

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

### Output

**Output reporter (S0-R)**

**Encoding:**

- \( a = CT \)GGCT, \( b = CGCAGC \)
- \( GGCT = 0, \) \( CAGC = 1 \)

---

[Benenson\&Shapiro, Nature, 2001]
Problems of the previous design

- Evidence of Ligase-free computation, but inefficient
  - Often *FokI* cuts only one input DNA strand
  - Computation stalled after a few steps

[Benenson & Shapiro, PNAS, 2003]
Molecular Computing Machine
Uses its Input as Fuel

PNAS, 2003

Kobi Benenson

Joint work with
Rivka Adar, Tamar Paz-Elizur, Zvi Livneh and Ehud Shapiro

Department of Computer Science and Applied Math &
Department of Biological Chemistry
Weizmann Institute of Science, Rehovot, Israel
Information destruction in electronic computers: bit reset to zero
(Landauer, Bennett)

Entropy decreasing and hence free energy-consuming operation, which is avoided in reversible computing
Information destruction in biology: physical degradation of the bit sequence (string to multiset)

$$xyz > 40kT$$

$${x, yz}$$

Entropy increasing and energy-releasing operation, which can be exploited to avoid the demand for external energy source.
• Input destruction can be used as a source of energy
• If output is smaller than input (e.g. yes/no questions), computation can be accomplished without external energy
• We realized this theoretical possibility
Finite automaton: an example

An even number of a’s

Two-states, two-symbols automaton

[S. Benenson & R. Shapiro, *PNAS*, 2003]
Automaton A1

An even number of a’s

\[
\begin{align*}
S0, a & \rightarrow S1 \\
S0, b & \rightarrow S0 \\
S1, a & \rightarrow S0 \\
S1, b & \rightarrow S1
\end{align*}
\]

[Benenson&Shapiro, PNAS, 2003]
An even number of a's

Automaton A1

[Benenson & Shapiro, PNAS, 2003]
Automaton A1

An even number of a’s

$S_0, a \rightarrow S_1$
$S_0, b \rightarrow S_0$
$S_1, a \rightarrow S_0$
$S_1, b \rightarrow S_1$

[Benenson & Shapiro, PNAS, 2003]
Automaton A1

An even number of a’s

\[
\begin{align*}
S0, a & \Rightarrow S1 \\
S0, b & \Rightarrow S0 \\
S1, a & \Rightarrow S0 \\
S1, b & \Rightarrow S1 \\
\end{align*}
\]

[Benenson & Shapiro, PNAS, 2003]
Automaton A1

An even number of a’s

\[
\begin{align*}
S0, a & \rightarrow SI \\
S0, b & \rightarrow SO \\
SI, a & \rightarrow SO \\
SI, b & \rightarrow SI
\end{align*}
\]

[Benenson & Shapiro, PNAS, 2003]
Automaton A1

An even number of a’s

\[ S_0, a \rightarrow S_1 \]
\[ S_0, b \rightarrow S_0 \]
\[ S_1, a \rightarrow S_0 \]
\[ S_1, b \rightarrow S_1 \]

[Benenson & Shapiro, PNAS, 2003]
Automaton A1

An even number of a’s

\[ S0, a \rightarrow S1 \]
\[ S0, b \rightarrow S0 \]
\[ S1, a \rightarrow S0 \]
\[ S1, b \rightarrow S1 \]

[Benenson & Shapiro, PNAS, 2003]
Previous molecular finite automaton

Encoding: $a = \text{CTGGCT}$, $b = \text{CGCAGC}$

$\text{GGCT} = 0$, $\text{CAGC} = 1$

A new molecular automaton

• Key differences:
  • No Ligase, hence no ATP
  • Software reuse – molecule not consumed during transition
• Hence a fixed amount of hardware and software molecules may process input of any length without external source of energy

[Benenson&Shapiro, PNAS, 2003]
A new molecular automaton

• Significant improvement of yields and performance

[Benenson & Shapiro, PNAS, 2003]
Modifications in the molecular design

Software is recycled

No Ligase – no ATP

[Benenson & Shapiro, PNAS, 2003]
Problems of the previous design

• Evidence of Ligase-free computation, but inefficient
  • Often FokI cuts only one input DNA strand
  • Computation stalled after a few steps

[Benenson & Shapiro, PNAS, 2003]
Modifications in the molecular design

Explanation of state and symbol encoding

<table>
<thead>
<tr>
<th>Symbol</th>
<th>a</th>
<th>b</th>
<th>terminator (t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>encodings &amp; &lt;state, symbol&gt; sticky ends</td>
<td>&lt;S1, a&gt;</td>
<td>&lt;S1, b&gt;</td>
<td>&lt;S1, t&gt;</td>
</tr>
<tr>
<td></td>
<td>TGGCT</td>
<td>GCAGG</td>
<td>GTCGG</td>
</tr>
<tr>
<td></td>
<td>&lt;S0, a&gt;</td>
<td>&lt;S0, b&gt;</td>
<td>&lt;S0, t&gt;</td>
</tr>
</tbody>
</table>

3-bp spacers between symbols

Symbols 5-bp long

[Benenson & Shapiro, PNAS, 2003]
Modifications in the molecular design

The software molecules

Shortest possible spacers between the FokI site and the \( <\text{state, symbol}> \) recognition sticky ends: 0-, 1- and 2-bp

[Benenson&Shapiro, PNAS, 2003]
Experimental implementation
The automata

A1: even number of a’s
A2: even number of symbols
A3: ends with b

The inputs

I1: abb
I2: abba
I3: babbabb
I4: babbabba
I5: baaaaabb
I6: baaaaabba
I7: abbbbbabbbabb
I8: abbbbaaaabba

[Benenson & Shapiro, PNAS, 2003]
Single step proof

Phosphorylated and non-phosphorylated single-symbol input

[Benenson&Shapiro, PNAS, 2003]
Single step proof

Phosphorylated and non-phosphorylated transition molecule (T1)

$T_a$

$32^P - A$

$12^P$

GGATGC

CCTACGCCGA - O - P

$T_b$

$32^P - A$

$12^P$

GGATGC

CCTACGCCGA - O - H

[Benenson & Shapiro, PNAS, 2003]
Single step proof

- All possible combinations are mixed with FokI (No Ligase and No ATP in all the reactions)
- We prove that there is no Ligase and ATP contamination in the FokI batch

[Benenson & Shapiro, PNAS, 2003]
Single step proof

[Benenson&Shapiro, PNAS, 2003]
Computation capabilities

A set of 8 inputs was tested with 3 software programs, at standard conditions:

4 µM FokI
4 µM software
1 µM input
8 °C
20 min

[Benenson&Shapiro, PNAS, 2003]
Computation capabilities

Direct output detection by denaturing PAGE

Automaton

Expected output S…

Input I…

S1  S0

1 0 0 1 0 1 1 0
1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0

S1  S0

1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0

1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8

[Benenson & Shapiro, PNAS, 2003]
Computation capabilities

- All the runs allowed correct major results with minor byproducts
- Only small ratio of the byproducts represent computation error

Automaton

Expected output S...

Input I...

\[\begin{array}{cccccccc}
S1 & S0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}\]

\[\begin{array}{cccccccc}
A1 & A2 & A3 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
\end{array}\]

\[\begin{array}{cccccccc}
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}\]

\[\begin{array}{cccccccc}
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}\]

\[\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}\]

[Benenson & Shapiro, *PNAS*, 2003]
Software recycling

- Automaton: A1
- Input: I8
- Each software molecule: 0.075 molar ratio to the input

T2, T5 and T8 performed on the average 29, 21 and 54 transitions each.

[Benenson & Shapiro, PNAS, 2003]
Optimization: the fastest computation

- 4 µM software, 4 µM hardware and 10 nM input
  - Rate: 20 sec/operation/molecule
- 50-fold improvement over the previous system

[Benenson&Shapiro, PNAS, 2003]
Optimization: the best parallel performance

• 10 μM software, 10 μM hardware and 5 μM input
• Combined rate: $6.646 \times 10^{10}$ operations/sec/μl
• ~8000-fold improvement over the previous system

[Benenson&Shapiro, PNAS, 2003]
Conclusions

Our experiments demonstrate:

- $3 \times 10^{12}$ automata/µl (240-fold improvement)

- Performing $6.6 \times 10^{10}$ transitions/sec/µl (8000-fold improvement)

- With transition fidelity of 99.9% (2-fold improvement)

- Dissipating $1.02 \times 10^{-8}$ W/µl as heat at ambient temperature

[Benenson & Shapiro, PNAS, 2003]
Conclusions

We developed a molecular finite automaton that realizes the theoretical possibility using the input as the sole source of energy

[Benenson&Shapiro, PNAS, 2003]
Stochastic computing with biomolecular automata,

PNAS, 2004

Yaakov Benenson & Ehud Shapiro
and sequences of the oligonucleotides were: A.cal.s (CAGGGCCGCAGGGCCGCCTGGCTGCAAAAAATTA-
CCGATTAAGTTGGA-FAM), A.cal.as (Cy5-CCAACTTAA-
TCGGTAA
TTTT
TGGCAGCCAGGCCCTGCGGCCCTGC-
GGC), B.cal.s (GGCTGCCTGGCGGCCTGGCTGCCGCA-
GGGCCAAAAATTACCGATTAGTTGG
A-FAM), B.cal.as
(Cy5-CCAACTTAATCGGTAA
TTTT
TGGCCCTGCGGCAGCCAGGCCGCCAGGC). 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 theT1-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
Concentration 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
Concentration of the corresponding transition was added. Thus,
the deterministic part of the computation was performed by a
total of 1.5
Mtransition mixture. For the last choices step, a
total of the 0.5
Mconcentration of the tested transition molecules mixture was taken at a required ratio.
FokI enzyme was added at the 2
Mfinal 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 af
3ha r
2hr
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. 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 con-
fignurations (state-input combinations) that arise during the computation and the sole transition that applies to each con-
fignuration. (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 transi-
tion 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 con-
fignurations 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 con-
fignuration in which the state-symbol combination is
S0,
S1,
and the two applicable
transitions are T3 and T4.
formed according to the Cy5 label (PMT 500–550 V) on the 5'H11032 terminus of the antisense strand, because it gave more reproducible and consistent results than the FAM label on the 3'H11032 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/FokI, respectively). Each pair of competing transition molecules was maintained at 0.5 Mc concentration. Computation reactions containing the input, transition molecules, and the FokI classII restriction enzyme (54 Ms stock; 60 units/l; New England Biolabs) were performed in 10 lof NEB4 buffer at 15°C. Before input addition reaction mixtures were preincubated with FokI at 15°C for 20 min. After 2 h, 2-l aliquots were taken from the reaction mixtures, added to 4 lof stop solution [9 volumes of Formamide (Merck) and 1 volume of 1089 mM Tris 89 mM boric acid 2m MEDTA, pH 8.3 (TBE)]. Half of the sample above was assayed by 15% denaturing...
An autonomous molecular computer for logical control of gene expression

Nature, 2004

Yaakov Benenson & Ehud Shapiro
Medicine in 2050: “Doctor in a Cell”

Programmable Computer

Molecular Input

Molecular Output

[Benenson & Shapiro, Nature, 2001]
**Figure 1** Logical design and logical operation of the molecular computer. 

**a**, Function and nodular organization of the molecular computer. 

**b**, Example diagnostic rules for simplified models of SCLC\(^1\)\(^9\) and prostate cancer\(^2\)\(^0\), indicating overexpression (↑) or underexpression (↓) of a disease-related gene. The first rule states that if genes *ASCL1*, *GRIA2*, *INSM1* and *PTTG1* are overexpressed then administer the ssDNA molecule TCTCCAGCGTGCCCAT (oblimersen), purported to be an antisense therapy drug for SCLC\(^2\)\(^7\). The second rule states that if the genes *PPAP2B* and *GSTP1* are underexpressed and the genes *PIM1* and *HPN* are overexpressed then administer the ssDNA molecule GTTGGTATTGCACAT, purported to be a drug for prostate cancer\(^2\)\(^5\). 

**c**, Transition diagram of the diagnostic automaton. 

**d**, The computation that diagnoses prostate cancer.
Molecular components and computational step of a molecular automaton

DNA 5bp recognition site

Hardware (FokI)

9nt

13nt

FokI recognition site

State-symbol recognition sticky end

State-symbol encoding sticky end

Current symbol

Next symbol

Input molecule

Hardware / Software / Input complex

Recycled Hardware / Software complex

Cleaved input symbol

Cleaved input molecule
a) Computation module: logical analysis of disease indicators

Yes, PPAP2B↓, GSTP↓, PIM↑, HPN↑ → Yes-verification → inactive drug

Yes, GSTP↓, PIM↑, HPN↑ → Yes

Yes, PIM↑, HPN↑ → Yes

Low Yes, PIM↑ → No

No, HPN↑ → No

Yes, HPN↑ → Yes

Low active Yes, PIM↑ → Yes

High active Yes, PIM↑ → Yes

Low inactive Yes, PIM↑ → Yes

High inactive Yes, PIM↑ → No

Low No, HPN↑ → No

High Yes, HPN↑ → Yes

High active drug

High administered drug

Low active drug suppressor

Low suppressed drug

MDM2 mRNA

MDM2 protein

b) Input module: software regulation by mRNA levels

Active Yes, PIM↑ → No

Inactive Yes, PIM↑ → Yes

PIM1 inactivation tag

PIM1 activation tag

PIM1 indicator

High active Yes, PIM↑ → Yes

Low active Yes, PIM↑ → No

High inactive Yes, PIM↑ → No

Low inactive Yes, PIM↑ → Yes
are compensated for by a similar change in absolute concentration of the transition molecules (Supplementary Fig. S3). Third, false-positive or false-negative diagnoses may be compensated for as explained above.

The operation of the computer modules was verified separately and jointly: transition regulation by the input module (Fig. 2b; see also Supplementary Fig. S2) was verified independently (Fig. 3a; see also Supplementary Fig. S3); the input and computation modules

Figure 2 Operation of the molecular computer. The complete sequences for all molecules shown are given in the Supplementary Methods. a, Part of the computation path for the diagnostic molecule for prostate cancer with all molecular indicators present, ending in drug release. The initial diagnostic molecule consists of a diagnosis moiety (grey) that encodes the left-hand side of the diagnostic rule (Fig. 1b) and a drug-administration moiety (light purple) incorporating an inactive drug loop (dark purple). At each computation step, the prevailing transition is shown, except for the processing of the symbol $PIM1 \uparrow$, for which details of the stochastic choice, accomplished by a regulated pair of competing transition molecules, are shown (dashed box, see c). b, Regulation of the two transitions for $PIM1 \uparrow$ by sub-sequences (tags) of overexpressed $PIM1$ mRNA, resulting in a relatively high level of the Yes $PIM1 \rightarrow$ Yes transition molecules and low level of the Yes $PIM1 \rightarrow$ No molecules. Each transition molecule contains regulation (green, orange) and computation (blue, grey) fragments. The ‘inactivation tag’ of $PIM1$ mRNA (light orange) displaces the 5’ $\rightarrow$ 3’ strand of the transition molecule Yes $PIM1 \rightarrow$ No and destroys its computation fragment. The ‘activation tag’ of $PIM1$ mRNA (light green) activates the transition molecule Yes $PIM1 \rightarrow$ Yes. Initially, a protecting oligonucleotide (green) partially hybridizes to the 3’ $\rightarrow$ 5’ strand of the transition molecule and blocks the correct annealing of its 5’ $\rightarrow$ 3’ strand. The ‘activation tag’ displaces the protecting strand, allowing such annealing and rendering an active Yes $PIM1 \rightarrow$ Yes transition. Ideally, one $PIM1$ mRNA molecule inactivates one Yes $PIM1 \rightarrow$ No and activates one Yes $PIM1 \rightarrow$ Yes transition molecule. c, Stochastic processing of the symbol $PIM1 \uparrow$ by a regulated pair of competing transition molecules. The probability of a Yes $\rightarrow$ Yes transition is high, resulting in a high level of diagnostic molecules in the state Yes and a low level in state No. d, Combining computation results for both types of diagnostic molecules, both with high Yes and low No final states, results in high release of drug and low release of drug suppressor, and hence in the administration of the drug.
overexpressed PIM1 mRNA,
resulting in a relatively high level of the Yes\textsuperscript{\textit{\textit{PIM1}}}\textsuperscript{\textit{Yes}} transition molecules
and low level of the Yes \textsuperscript{\textit{\textit{PIM1}}}\textsuperscript{No} molecules.
Input module: software regulation by mRNA levels

Active Yes $PIM1^{\uparrow}$ No

Inactive Yes $PIM1^{\uparrow}$ Yes

$PIM1$ inactivation tag $+$ $PIM1$ activation tag

Check for $PIM1^{\uparrow}$ indicator

High active Yes $PIM1^{\uparrow}$ Yes

Low inactive Yes $PIM1^{\uparrow}$ Yes

Low active Yes $PIM1^{\uparrow}$ No

High inactive Yes $PIM1^{\uparrow}$ No
are compensated for by a similar change in absolute concentration of the transition molecules (Supplementary Fig. S3). Third, false-positive or false-negative diagnoses may be compensated for as explained above.

The operation of the computer modules was verified separately and jointly: transition regulation by the input module (Fig. 2b; see also Supplementary Fig. S2) was verified independently (Fig. 3a; see also Supplementary Fig. S3); the input and computation modules

![Diagram](image)

**a** Computation module: logical analysis of disease indicators

- Yes, **PPAP2B**↓ **GSTP1**↓ **PIM1**↑ **HPN**↑ Yes-verification Inactive drug
- Yes **PPAP2B**↓ Yes
- Yes, **GSTP1**↓ **PIM1**↑ **HPN**↑
- Yes **GSTP1**↓ Yes
- Yes, **PIM1**↑ **HPN**↑
- Low Yes **PIM1**↑ No
- No, **HPN**↑
- High Yes **PIM1**↑ Yes
- Yes **HPN**↑ Yes
- No **HPN**↑ No
- Low No
- High Yes
- Yes **HPN**↑ Yes
- Yes **HPN**↑ Yes
- High active drug

**c** Probabilistic check

- Yes, **PIM1**↑ **HPN**↑
- Low Yes **PIM1**↑ No
- High Yes **PIM1**↑ Yes
- Yes **HPN**↑ Yes
- Yes **HPN**↑ Yes
- High active drug
a Diagnostic molecules for prostate cancer

b Transition molecule for \( \text{Yes} \rightarrow \text{Yes} \)

c Transition molecule for \( \text{Yes} \rightarrow \text{No} \)

d Transition regulation by point mutation
indicators. The location of the initial, intermediate and output molecules is indicated on the result of a diagnostic computation for the indicated combination of molecular automata with the diagnostic rules for SCLC and prostate cancer (PC). Each lane shows F, carboxyfluorescein; R, tetramethyl rhodamine; Y, Cy5.

inactive states, the relative concentration of which is regulated by mRNA concentration. Shows positive (blue and pink) and negative (green) transition, each in both active and inactive states.

Figure 3

Experimental demonstration of diagnosis. a, Regulation of competing transitions by pTRI-Xef mRNA (bar chart) representing an example molecular indicator. The gel shows positive (blue and pink) and negative (green) transition, each in both active and inactive states, the relative concentration of which is regulated by mRNA concentration.
b, Validation of the diagnostic automata with the diagnostic rules for SCLC and prostate cancer (PC). Each lane shows the result of a diagnostic computation for the indicated combination of molecular indicators. The location of the initial, intermediate and output molecules is indicated on the left and a predicted trace of the diagnostic computations is shown on both sides. Some, but not all, intermediates are visible owing to their incomplete processing by the

The regulation of transition molecules by mRNA (Fig. 2b) was demonstrated (Fig. 3a) with an mRNA transcript of about 1900 nucleotides as an example indicator. A correlation between the ssDNA oligonucleotides to represent disease-related mRNA and changing the absolute concentration of competing transitions to extend this approach to detect insertion and deletion mutations.

In diagnosing a disease model with multiple indicators we used up to four molecular indicators, although the specific symbol encoding used can provide up to eight string letters to nature.

In diagnosing a disease model with multiple indicators we used the diagnostic string PTTG1↑ CDKN2A↑ and the two indicators of prostate cancer checked by the string PIM1↑ HPN↑. The gene CDKNA2 is another indicator for SCLC when overexpressed, and it is used here for technical reasons. The presence of indicators for each disease model as well as the expected diagnostic output by each automaton are indicated above the lanes.
Diagnostic output

- Cross hybridization product
- Unprocessed diagnostic molecule
- Active drug
- Active drug suppressor

Y N

Drug/drug suppressor hybrid