Chemical implementation of finite-state machines

ALLEN HJELMFE1, EDWARD D. WEINBERGER†, and JOHN ROSS‡

†Max-Planck-Institut für Biophysikalische Chemie, D-3400 Göttingen, Federal Republic of Germany; and ‡Department of Chemistry, Stanford University, Stanford, CA 94305

Contributed by John Ross, September 23, 1991

ABSTRACT With methods developed in a prior article on the chemical kinetic implementation of a McCulloch-Pitts neuron, connections among neurons, logic gates, and a clocking mechanism, we construct examples of clocked finite-state machines. These machines include a binary decoder, a binary adder, and a stack memory. An example of the operation of the binary adder is given, and the chemical concentrations corresponding to the state of each chemical neuron are followed in time. Using these methods, we can, in principle, construct a universal Turing machine, and these chemical networks inherit the halting problem.

In a prior article (1) we discussed the implementation of a chemical neural network: we wrote a reaction mechanism with stationary-state properties of a McCulloch-Pitts neuron (2, 3) and developed chemical interneuronal connections, basic logic gates, a clocking mechanism, and input and output of the entire network. In this article we combine these chemical components to construct three devices: a binary decoder, a binary adder, and a stack memory. The method of construction can be used to make the finite-state component of a universal Turing machine (4–6), as any finite-state machine can be simulated by clocked neural networks (5). In principle, by coupling this particular finite-state machine with a readable–writable tape, such as a polymer like DNA or a pair of stack memory devices, the chemical implementation of a universal Turing machine based on kinetic reaction mechanisms is realizable.

We leave for later study a related issue: given a biological (chemical) reaction mechanism what logical operations, what computations, can this mechanism perform for given inputs.

We begin with a brief review of the components of a chemical neural network, and then we discuss the construction of a binary adder and a stack memory.

Construction of Chemical Neural Networks

A Single Chemical Neuron. As a basis for a “chemical neuron” we choose a cyclic enzyme mechanism studied by Okamoto et al. (7, 8):

$$I_1^* + C_i \rightleftharpoons X_{11} + C_i \quad J_{1i} = k_1C_i - k_{-1}C_iX_{1i}$$

$$X_{1i} + B_i \rightleftharpoons X_{2i}^* + A_i \quad J_{2i} = k_2X_{1i}B_i - k_{-2}A_i$$

$$X_{3i} + A_i \rightleftharpoons X_{4i}^* + B_i \quad J_{3i} = k_3X_{3i}A_i - k_{-3}B_i$$

$$X_{3i}^* \rightleftharpoons I_{2i}^* \quad J_{4i} = k_4X_{3i}^* - k_{-4}$$

where the concentrations of the species marked by the (*) superscript are held at a constant value, either by buffering or by flows, and have been absorbed into the rate constants, $A_i$ and $B_i$ are the state species and are related by a conservation constraint, $A_i + B_i = A_0$. The stationary-state concentrations are functions of the concentration of the catalyst $C_i$. By using the rate constants given in ref. 1, the stationary-state concentration of $A_i$ is $<2 \times 10^{-4}$ mmol/liter and of $B_i$ is $>0.999$ mmol/liter for $C_i < 0.90$ mmol/liter, and the concentration of $A_i$ is $>0.999$ mmol/liter and of $B_i$ is $<2 \times 10^{-4}$ mmol/liter for $C_i > 1.10$ mmol/liter. Thus, the chemical neuron has two states, and the concentration of $C_i$ determines the state of neuron $i$.

Clocking. In the neural networks we describe here the state of a chemical neuron is allowed to change only at discrete times. This discreteness of time and synchronization of state changes can be implemented chemically by the use of an autonomously oscillating catalyst $\varepsilon$. We assume that $\varepsilon$ oscillates in a nonsinusoidal manner, as is common in many chemical oscillators (9). The concentration of $\varepsilon$ is assumed to be very small, except during an interval short compared with the oscillator period and with the relaxation time of a chemical neuron (Eq. 1). The catalyst $\varepsilon$ interacts with the species $A_j$ (or $B_j$) of each neuron $j$.

$$A_j \rightleftharpoons A_j^* \quad B_j \rightleftharpoons B_j^*$$

and rapid equilibration occurs only during the short time interval when the concentration of $\varepsilon$ is large. In Fig. 1 we show schematically the time variation of the concentrations of $A_i$ and $A_i^*$ as determined by the concentration of $C_i$. $A_i$ is the state of neuron $i$ at a given time, say $t = 0$ for the interval 0 to 1 in Fig. 1 and determines $A_j^*$ in the next time interval, 1 to 2. The state of neuron $j$ at time $t - 1$ determines the state of neuron $i$ at time $t$. Thus, the $A_j^*$ at time $t$ determines the state of neuron $i$ at time $t$.

Interneuronal Connections. The effect of the state of the other neurons $j, k, \ldots$ on neuron $i$ is expressed in $C_i$. The species $A_j^*, \ldots$ or $B_j^*, \ldots$ affect the concentration of the catalyst $C_i$ by activation reactions.

$$E_j^* + A_j^* \rightleftharpoons C_{ij} \quad C_{ij} = \frac{E_j}{1 + \frac{E_j}{K_{ij}}}$$

where $E_j^*$ is the concentration of the pulse of the catalyst $\varepsilon$ and to be fast compared with the time scale of mechanism 1. The sum of the active forms of the enzyme determines $C_i$:

$$C_i = \sum_j C_{ij}$$

In Fig. 2 we show schematically the influence of neurons $j$, $k$, and $l$ on neuron $i$. The state of neuron $j$ determines the concentration of $C_{ij}$, and the firing of neuron $i$ is inhibited by the firing of neuron $j$ (Eq. 4). The states of neurons $k$ and $l$ (not
shown) likewise determine the concentrations $C_{ik}$ and $C_{ij}$, and the sum of $C_{ij}$, $C_{ik}$, and $C_{ij}$ is $C_i$, the parameter that determines the state of neuron $i$. The state of neuron $i$ determines the concentration of $C_{ik}$, and the firing of neuron $k$ is excited by the firing of neuron $i$.

The combination of reactions 3 and 4 determines the logical operation of neuron $i$ on the states of neurons $j, \ldots$. That is, the state of neuron $i$ at time $t$ is determined by a logical operation on the states of neurons $j, \ldots$ at time $t-1$. In ref. 1 we describe how various logical operations can be represented, such as AND, OR, NOR, etc.

We also use a connection where the connection enzyme ($C_{ij}$) in Eq. 5 is inhibited or activated by more than one species. $A_j$ and $A_k$ interact with the same enzyme $E_j$.

$$E_j + A_j := C_{ij},$$

$$E_j + A_k := (E_jA_k),$$

and

$$C_j = \frac{E_j^2 + 1}{1 + \frac{1}{K_AA_j^2}},$$

where $E_j = E_j + C_j + (E_jA_k)$, $K_A$ is the equilibrium constant of the activation reaction (Eq. 6), and $K_1$ is the equilibrium constant of the inhibition reaction (Eq. 7). These reactions allow specific inhibition of one connection, instead of the nonspecific inhibition given by Eq. 4.

Examples of Finite-State Machines

One copy of the basic reaction mechanism of a neuron (Eq. 1) exists for each chemical neuron in the network. Each neuron is chemically distinct, but for convenience we assume that the reactions that constitute each neuron are mechanistically similar. A machine is specified by the number of neurons, the form of the connections between the neurons (Eqs. 3, 4, or 8), which neurons represent the output of the machine, and which concentrations represent the input to the machine.

Binary Decoder. The first device we construct is a binary decoder composed of four neurons ($i = 3-6$) and two input concentrations $A_j(t)$ and $A_k(t)$, which we assume to be controlled by the external world, which are represented here as the state species of neurons 1 and 2. A binary number is represented as a string of digits presented to the machine sequentially in time with the least-significant digit first. $A_j(t)$ and $A_k(t)$ are each digits of such numbers. These numbers are presented, digit by digit in parallel, to the binary decoder.

![Diagram](image-url)

FIG. 2. Schematic of two reaction mechanisms constituting neurons $i$ and $j$ and the influence of neurons $j$, $k$, and $l$ on neuron $i$. All reactions are reversible. The firing of neuron $j$ inhibits the firing of neuron $i$, and neurons $k$ and $l$ (not shown) also influence the state of neuron $i$. The firing of neuron $i$ inhibits the firing of neuron $k$.

which causes one and only one of the neurons with $i = 3-6$ to fire at time $t + 1$. The catalyst concentrations of the four neurons are given by

$$C_3 = \frac{1}{1 + \frac{1}{24_1} + \frac{1}{24_2}}, \text{ neuron 3 fires only if } A_1 = 1, A_2 = 1,$$

$$C_4 = \frac{1}{1 + \frac{1}{24_1} + \frac{1}{24_2}}, \text{ neuron 4 fires only if } A_1 = 1, A_2 = 0,$$

$$C_5 = \frac{1}{1 + \frac{1}{24_1} + \frac{1}{24_2}}, \text{ neuron 5 fires only if } A_1 = 0, A_2 = 1,$$

and

$$C_6 = \frac{1}{1 + \frac{1}{24_1} + \frac{1}{24_2}}, \text{ neuron 6 fires only if } A_1 = 0, A_2 = 0.$$
Fig. 3. Schematic of the neurons and connections in the binary decoder and adder. The half-shaded circles denote neurons. The connection emerging from the shaded side is the output (state) of the neuron. Connections entering the unshaded provide input to the neuron: — , excitatory connections; — O, inhibitory connections. The number of firing excitatory inputs is summed, and when that number is greater than the number in the neuron and no inhibitory inputs are firing, the neuron fires. In this notation neuron 6 is a NOR gate, neuron 5 is an A2 AND NOT A1 gate, and neurons 3 and 7–12 are AND gates. Some of the connections are denoted by broken lines for clarity. Each column of the adder represents one state of the adder; carry 0 or carry 1; and each row represents one of the input combinations: [1 1], [1 0], [0 0]. Thus each neuron in the adder portion represents one row in Table 1.

represents one of the machine states and each row represents one of the possible inputs to the adder. In Fig. 3, neurons 7–9 represent the carry-0 state, and neurons 7 and 10 represent the input [0 0]. Each neuron i in the adder represents line i of Table 1. At any given time exactly one neuron in the adder is firing. Each neuron in the adder excites the neurons in the column representing its state at t + 1 (determined from Table 1), and each input to the adder from the decoder excites the neurons in one row. Thus, only one neuron in the adder has two firing inputs at any given time. Because all the neurons of the adder are AND-neurons, only one neuron will fire at time t + 1. This one firing neuron also gives the output of the adder, which is determined by Table 1.

To follow the rules of Table 1 we choose the catalyst concentrations for the six AND-neurons that compose the binary adder as follows:

\[
C_7 = \frac{1}{1 + \frac{1}{2A_5} + \frac{1}{1 + \frac{1}{2A_{11}} + \frac{1}{1 + \frac{1}{2A_{12}}}}}, \quad [13]
\]

\[
C_8 = \frac{1}{1 + \frac{1}{2A_5} + \frac{1}{1 + \frac{1}{2A_{11}} + \frac{1}{1 + \frac{1}{2A_{12}}}}}, \quad [14]
\]

\[
C_9 = \frac{1}{1 + \frac{1}{2A_5} + \frac{1}{1 + \frac{1}{2A_{11}} + \frac{1}{1 + \frac{1}{2A_{12}}}}}, \quad [15]
\]

\[
C_{10} = \frac{1}{1 + \frac{1}{2A_5} + \frac{1}{1 + \frac{1}{2A_{11}} + \frac{1}{1 + \frac{1}{2A_{12}}}}}, \quad [16]
\]

\[
C_{11} = \frac{1}{1 + \frac{1}{2A_5} + \frac{1}{1 + \frac{1}{2A_{11}} + \frac{1}{1 + \frac{1}{2A_{12}}}}}, \quad [17]
\]

\[
C_{12} = \frac{1}{1 + \frac{1}{2A_5} + \frac{1}{1 + \frac{1}{2A_{11}} + \frac{1}{1 + \frac{1}{2A_{12}}}}}, \quad [18]
\]

where \(A_i\) is the state species of the neuron corresponding to \(C_i\) and for \(i < 5\), these are the neurons of the previous subsection (the binary decoder). As indicated in Table 1 the input of [0 1] and [1 0] is degenerate. Thus, the decoder neurons 4 and 5 both excite the same row (neurons 8 and 11 in Fig. 3) of the AND-neuron matrix.

The operation of the decoder and adder is illustrated in Fig. 4, where we plot the time evolution of the state species \(A_i\) concentrations. At \(t = 0\) the two binary digits \(A_1(0) = 1\) and \(A_2(0) = 0\) are presented to the decoder. At \(t = 1\) neuron 4 of the decoder, which fires if and only if \(A_1 = 1\) and \(A_2 = 0\) (Eq. 10), fires and excites one row of neurons, neurons 8 and 11, in the binary adder. Also at time \(t = 1\), neuron 7 of the adder is firing, and it is an input to the carry-0 column (neurons 7–9) of the adder. At \(t = 2\), only neuron 8 of the adder has two firing inputs, so in the adder only neuron 8 fires. From Table 1 neuron 8 signifies that the adder outputs a 1 and carries a 0. The output of the adder lags behind the input to the decoder by two time steps. The first two digits of output \((t = 1, 2)\) are discounted bits because of the time lag. The relevant output starts at \(t = 3\). Likewise, the last two input digits are not part of the output due to the time lag.

Stack. The last clocked device to be described is a first-in last-out stack memory. A finite-state machine augmented with two infinite stacks is equivalent in power to a Turing machine with one infinite tape (4); it is computationally universal.

The typical example of a stack is a stack of plates on a spring, and the spring pushes the plates up so that only one plate is visible. If the plates represent data (binary digits for example), then a particular data item can only be reached by removing all the plates above it. Following the analogy, a stack can be imagined as a linear array of neurons extending downward from a top neuron. Each neuron, Eq. 1, is coupled

<table>
<thead>
<tr>
<th>(i)</th>
<th>(A_1)</th>
<th>(A_2)</th>
<th>(S)</th>
<th>(S_{r+1})</th>
<th>(O)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1*</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1*</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\(A_1\) and \(A_2\) are the two input digits, \(S\) and \(S_{r+1}\) are the machine states at time \(t\) and \(t + 1\), and \(O\) is the output digit. Each neuron \(i\) (7–12) in the adder corresponds to the same indexed row \(i\) in the table.

*The case for \(A_1 = 1\) and \(A_2 = 0\) is equivalent.
to its two neighbors, and only the top neuron can be read or modified by an external finite-state machine, as for example the binary adder in the previous section. At each time step the stack can perform one of three operations based on the command received from the external finite-state machine: "remember," "pop," and "push." For the remember operation none of the neurons in the stack change state. For the pop operation, each neuron transfers its state to the neuron above it in the stack, and the state of the top neuron is transferred to the external finite-state machine. For the push operation, each neuron transfers its state to the neuron below it in the stack, and the external finite-state machine transfers information to the top neuron.

Pictured in Fig. 5 is a stack consisting of four neurons ($i = 1-4$). Neurons 5 and 6 (not shown) are part of an external finite-state machine and determine the operation implemented by the stack. Neuron 7 is also part of the external finite-state machine and determines the input operation $A_1$ accepts data from $A_7$. To allow neurons 5 and 6 to control the stack we use the type of connections given by Eqs. 6-8. In Fig. 5 o denotes a connection that is excited by neuron 5 (i.e., the $B_3$ participates in reaction 7), □ denotes a connection excited by neuron 6, and □ denotes a connection that is inhibited by both neuron 5 and neuron 6. The catalyst concentrations of the four stack neurons are chosen as follows:

$$C_1 = \frac{1}{1 + \frac{1 + 50A_6'}{2A_1'}} \frac{1}{1 + \frac{1 + 50A_6'}{2A_1'}}$$

[19]

$$C_2 = \frac{1}{1 + \frac{1 + 50A_6'}{2A_1'}} \frac{1}{1 + \frac{1 + 50A_6'}{2A_1'}}$$

[20]

$$C_3 = \frac{1}{1 + \frac{1 + 50A_6'}{2A_1'}} \frac{1}{1 + \frac{1 + 50A_6'}{2A_1'}}$$

[21]

$$+ \frac{2}{1 + \frac{1 + 50(1 - A_3')}{2A_1'}} + \frac{2}{1 + \frac{1 + 50(1 - A_6')}{2A_1'}}$$

[22]

$$C_4 = \frac{1}{1 + \frac{1 + 50A_6'}{2A_1'}} \frac{1}{1 + \frac{1 + 50A_6'}{2A_1'}}$$

The concentrations $A_5$-$A_7$ are controlled by the external processor, and we take them as given. Consider the three stack operations as determined by the external control neurons $A_5$ and $A_6$. When $A_5 = A_6 = 0$, the last two terms are always small, and the magnitude of the first two terms depends on $A_1$ for all $C_i$: $C_i \approx \frac{5}{2}$ when $A_1 = 1$ or 0 when $A_1 = 0$. Thus, the state of neuron $i$ at time $t + 1$ is the same as its state at time $t$, and $A_5 = A_6 = 0$ causes the remember operation. When $A_1 = 1$ and $A_6 = 0$, the first and last terms are always small for all $C_i$ values. The second term is either 0 or $\frac{5}{2}$, depending on $A_i$, and the third term is either 0 or $\frac{5}{2}$, depending on the state of the next neuron higher in the stack. Thus, the state of neuron $i$ at time $t + 1$ is wholly determined by the state of the next neuron higher in the stack, and $A_5 = 1$ and $A_6 = 0$ cause the push operation. If $A_5 = 0$, $A_6 = 1$, the second and third terms are always small for all $C_i$. The first term is either 0 or $\frac{5}{2}$, depending on $A_i$, and the fourth term is either 0 or $\frac{5}{2}$, depending on the state of the neuron lower in the stack. Thus, the state of neuron $i$ at time $t + 1$ is wholly determined by the state of the next neuron lower in the stack, and $A_5 = 0$ and $A_6 = 1$ causes the pop operation.
Binary Adder.

Fig. 5. Schematic of the neurons and connections in the stack memory. The control neurons 5 and 6 are not shown because they are assumed to be controlled by the external world. Neurons 5 and 6 affect the connections between the neurons in the stack through the mechanism of Eqs. 6–8. We show these effects by symbols: neuron 6 excites connections marked by © and when neuron 6 is firing, this connection affects the state of neuron i; neuron 5 excites connections marked by ⊕, and when neuron 5 is firing, this connection affects the state of neuron i. Both neurons 5 and 6 inhibit connections marked by □, and when either neuron 5 or neuron 6 is firing, this connection does not affect the state of neuron i.

Conclusion

We have constructed a chemical kinetic system that can perform a programmed computation. We have demonstrated only simple computations, where both the computation and the underlying chemical dynamics are easily understood. In principle, a universal Turing machine can be constructed from two infinite chemical stacks and a neural network of the general form discussed in the sections on Binary Decoder and Binary Adder. Computational systems may, however, show much more complex behavior. Computation theory encompasses the possibility of computations with dynamical behavior that shows unpredictability stronger than "deterministic chaos." This unpredictability is due to Turing's halting problem (3), which states that it is unpredictable, without direct simulation, whether any arbitrary program will halt or attain a solution in finite time. The dynamical manifestation of unpredictability is a question about the existence and domain of basins of attraction (10). Computations may be viewed as the transient relaxation to a steady state, where the steady state represents the solution. Computationally powerful systems must be able to support arbitrarily (and unpredictably) long transients. The halting problem implies that direct simulation is the only general procedure to determine whether the transients will ever decay to a stationary state; in finite time an answer is not guaranteed. This unpredictability is stronger than that of deterministic chaos, where the motion is confined, at least, to an attractor (10).

Turing's theory uses devices with an infinite memory; however finite systems can show a related behavior termed "computational irreducibility" (11). In computationally irreducible bounded systems a solution is reached in finite time, but direct simulation is the most efficient deterministic method of solution. No more sophisticated deterministic method of solution exists and thus no computational shortcut exists to determine the final state of the system. Such considerations of computational irreducibility may apply to biological networks.

In our discussion of the operation of these neural networks we have assumed that any stochastic fluctuations do not interfere with the computation. However, stochastic fluctuations do occur. In the clocked networks we described, noise may significantly affect the integrity of the computation. Similar problems have been dealt with in the design of digital computers (3).

A.H. and E.D.W. thank Dr. Manfred Eigen for providing a stimulating research environment and for his interest in this work; these authors acknowledge support from the Max Planck Society. This work was also supported, in part, by the National Science Foundation.