Training an asymmetric signal perceptron through reinforcement in an artificial chemistry

Peter Banda¹, Christof Teuscher² and Darko Stefanovic³,⁴

¹Department of Computer Science, Portland State University, PO Box 751, Portland, OR 97207, USA
²Department of Electrical and Computer Engineering, Portland State University, Portland, OR, USA
³Department of Computer Science, and ⁴Center for Biomedical Engineering, University of New Mexico, Albuquerque, NM, USA

State-of-the-art biochemical systems for medical applications and chemical computing are application-specific and cannot be reprogrammed or trained once fabricated. The implementation of adaptive biochemical systems that would offer flexibility through programmability and autonomous adaptation faces major challenges because of the large number of required chemical species as well as the timing-sensitive feedback loops required for learning. In this paper, we begin addressing these challenges with a novel chemical perceptron that can solve all 14 linearly separable logic functions. The system performs asymmetric chemical arithmetic, learns through reinforcement and supports both Michaelis–Menten as well as mass-action kinetics. To enable cascading of the chemical perceptrons, we introduce thresholds that amplify the outputs. The simplicity of our model makes an actual wet implementation, in particular by DNA-strand displacement, possible.

1. Introduction

Learning and adaptation, along with homeostasis and growth, are among the fundamental characteristics that life in the most general sense exhibits [1,2]. They represent the ability of an individual to alter its response and decision-making by using feedback from the environment. This ability lets individuals adjust and escape predefined behavioural patterns given by evolution, i.e. adaptation at the level of the population.

In complex biological organisms, learning is carried out by neurons organized into networks and ultimately brains. Artificial neural network theory [3] investigates such systems by applying the circuit-based abstraction, wherein inputs and so-called synaptic weights are integrated and processed by the activation function to produce the output signal (an action potential). The question we ask is whether a system at the molecular level, an order of magnitude simpler than those of a neurobiological origin, could also adapt. Chemistry acts on different premises from formal neural circuits, e.g. it lacks topology, and its dynamics preserve matter.

In a previous paper [4], we demonstrated that an artificial chemical system can, in fact, perform autonomous learning. This feature has been absent from both theoretical and experimental chemistry research. Although neural network implementations using chemical media have been proposed and realized [5–10] in those systems, the learning part has been performed exclusively by an external (non-chemical) system that calculated adapted concentrations [10,11]. Beyond neural-networks approaches, in a theoretical paper, we proposed a schema for rudimentary learning within enzymatic chemistry [12], equivalent to non-negative least-squares regression, and in the laboratory, we exhibited rote learning of small decision trees [13] (training by example), but true learning calls for generalization. Moreover, a system based on gene regulatory networks embedded in \textit{Escherichia coli} demonstrated that even single-celled organisms could carry out associative learning, and so learning is substrate-free and universal [14].
Why look for a chemical system capable of learning? We have two motivations. First, the chemical medium is the basis for everything living on this planet; therefore, it is the most natural choice for implementing an artificial life, which will require learning. Second, a practical motivation is to introduce a flexible template for synthetic biochemistry that could find use in a variety of applications, such as drug delivery [15], pattern recognition and chemical computing [16]. Currently, the design of chemical systems is time-consuming and costly owing to the complexity of molecular interactions. Often, the only methodology available is a tedious trial-and-error approach. We argue that by understanding and developing adaptation in chemistry, we will be in a position where instead of multiple systems with hard-wired purpose we may design a single programmable template that can be trained (and retrained) for a desired functionality.

This paper extends the original concept of an autonomous chemical learning system, which we call a chemical perceptron, with the aim to simplify and reduce the number of reactions, such that an implementation in real chemistry becomes realistic. Our previous work [4] presented two variants of a two-input binary perceptron modelled in artificial chemistry—the weight-loop perceptron (WLP) and the weight-race perceptron (WRP). They represent two strategies for integrating the input and weight species. The first is based on a direct transformation of weights to the output, and reconstructing them after the processing; the second applies indirect comparison of weights by letting them race on the input-to-output reactions as catalysts. In both cases, learning is triggered by a discrepancy between the desired and the actual output species, where the concentration of the desired output is additively or subtractively combined with the concentrations of the weights. We demonstrated that the WLP and the WRP are correct proof-of-concept models for a chemical perceptron. However, the number of reactions and their complexity made them impractical for real chemistry implementation. The underlying cause of that complexity is a crucial characteristic they share—a representation symmetry of the species that encode the formal variables. Namely, all real-valued variables require a positive and a negative variant of a species; and, similarly, two species, the zero and one variants, represent each binary variable.

The overall design and functioning of the new asymmetric signal perceptron (ASP) substantially differ from those of the WLP and the WRP. The main improvement is the abolition of representation symmetry, which reduces the number of reactions by half. Furthermore, we introduce a special species, the input (clock) signal, that is provided alongside the regular input. The ASP determines the output value by thresholding, as opposed to a comparison of positive and negative output species concentrations. The thresholding is either imposed by an external observer (passive thresholding), or is implemented fully in chemistry (active thresholding). A variant of the ASP with active thresholding (TASP) can support modularization and cascading of multiple perceptrons. The ASP contains no inhibition and uses at most one catalyst per reaction. We revised the learning mechanism and introduced adaptation by the biologically more plausible reinforcement method [17,18]. More specifically, we train the ASP by injecting a penalty signal when it produces an incorrect output. ASP is compatible with Michaelis–Menten [19,20] as well as mass-action kinetics [21,22], hence it provides a universal description that all chemists understand and consequently can translate to the implementation substrate of their choice, such as DNA hybridization [23,24], or deoxyribozymes [25–27]. The major saving in design is offset by reduced robustness with respect to variation of kinetic rates, which is, however, still sufficiently high to mitigate the difficulties of precise reaction timing for real chemical implementations. On the other hand, the new model maintains the high performance of our previous models: it learns all 14 linearly separable binary functions with a 99.3–99.99% success rate.

### Table 1. The relation of an input pair $x_1$ and $x_2$ to output $y$ in the two-input binary perceptron, where $w_0$, $w_1$, and $w_2$ are weights, and $\theta$ is a threshold.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$w_0 &gt; \theta$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$w_0 + w_1 &gt; \theta$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$w_0 + w_2 &gt; \theta$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$w_0 + w_1 + w_2 &gt; \theta$</td>
</tr>
</tbody>
</table>

2. Two-input binary perceptron

The perceptron, introduced by Rosenblatt [28], is an early type of artificial neural network [29]. Neural network theory formalizes the functioning of biological neurons as linear-integration circuits with a threshold, sigmoid or other monotone activation function. Despite all simplifications, the perceptron is capable of non-trivial learning and forms the basis for more complex feed-forward neural networks. Here, we model the two-input binary perceptron with a threshold activation function that outputs one if the inner product of the weight vector and the input vector, $w_0 + w_1x_1 + x_2w_2$, is greater than the threshold $\theta$, and zero otherwise. Because the input is binary, the linear integration collapses to the four cases summarized in table 1. The two-input binary perceptron can learn all 14 linearly separable binary functions, i.e. all two-input binary functions except XOR and XNOR.

2.1. Learning by reinforcement

Classical perceptron learning [29], a type of supervised Hebbian learning [30], assumes that each training sample is presented as a pair comprising the training inputs (a vector $x$) and the desired output (a scalar $d$). If in the current state of the perceptron (i.e. its current weights), there is a discrepancy between its actual output $y$ and the desired output $d$, the error is fed back and triggers an adaptation of the participating weights. The adaptation of a weight $w_i$ for the training sample $(x_i,d)$ at time $t$ is defined as $w_i(t+1) = w_i(t) + \alpha(d - y(t))x_i$, where $\alpha \in (0, 1]$ is the learning rate.

An alternative learning method widely used for agent-based modelling is reinforcement learning [17]. In reinforcement learning, agents learn their expected behaviour from the consequences of their actions through rewards (positive reinforcements) and/or penalties (negative reinforcements). To replace the classical perceptron learning algorithm with reinforcement specified as a single penalty signal, the adaptation of weight $w_i$ for the penalized perceptron is $w_i(t + 1) = w_i(t) \pm bx_i$, where $b = 1$ is the constant penalty signal. The perceptron itself must determine whether the
weight should be increased or decreased as a consequence of penalization, i.e. the production of incorrect output. We decided to use reinforcement learning, because it is more biologically plausible [31] and it fits better to our asymmetric chemical implementation, as we will discuss in §3.3.

3. Model and design

We now describe a novel chemical implementation of the two-input binary perceptron—the ASP. To avoid confusion, we write the variables of the formal perceptron of §2 in lower case and the molecular species that encode them in upper case. Note that the correspondence between the two groups of symbols is not one-to-one. In fact, the model and the learning set-up of our chemical perceptrons do not adhere to those of the formal perceptron. For instance, instead of computing the linear sum of the participating weights, we let the weight species compete in a nonlinear manner. Furthermore, the weight contribution is not uniform, because the weight race favours one weight over the other. Despite the implementation differences and the inherent differences between circuit and chemical modelling primitives, our chemical model still qualitatively resembles its formal counterpart and is functionally equivalent to it.

To represent our model at the symbolic level, we use an artificial chemistry [32]. Here, the species are not assigned a molecular structure yet, and interact on the basis of stipulated chemical modelling primitives, our chemical model still qualitatively resembles its formal counterpart and is functionally equivalent to it.

To represent our model at the symbolic level, we use an artificial chemistry [32]. Here, the species are not assigned a molecular structure yet, and interact on the basis of stipulated chemical modelling primitives, our chemical model still qualitatively resembles its formal counterpart and is functionally equivalent to it.

3.1. Species for the representation of inputs, outputs and weights

Here, we describe the species used in the ASP design, highlighting the improvements with respect to the earlier WRP design.

(i) Input species and the clocked representation

The two-input formal binary perceptron introduced in §2 accepts four possible inputs \((x_1, x_2) \in \{(0, 0), (1, 0), (0, 1), (1, 1)\}\). In our artificial chemistry, we represent the presentation of an input as the injection of a molecular species into the reaction chamber, such that each formal variable with its associated value maps with one or several molecular species.

(ii) Output species and output interpretation

For output interpretation, we must carry out a reverse translation: we map the concentrations of the designated output species to the formal binary variable \(y\). The WRP has two complementary output species, \(Y^0\) and \(Y^1\), which are mutually exclusive, so if they occur simultaneously, they annihilate. We interpret the output as one if the concentration

\[
\begin{array}{|c|c|c|c|}
\hline
\text{binary input} & \text{WRP} & \text{ASP} & \text{minimal} \\
\hline
(x_1, x_2) = (0, 0) & \{X_1^0, X_2^0\} & \{S_{in}\} & 0 \\
(x_1, x_2) = (1, 0) & \{X_1^0, X_2^0\} & \{S_{in}, X_1\} & \{X_1\} \\
(x_1, x_2) = (0, 1) & \{X_1^0, X_2^0\} & \{S_{in}, X_2\} & \{X_2\} \\
(x_1, x_2) = (1, 1) & \{X_1^1, X_2^1\} & \{S_{in}, X_1, X_2\} & \{X_1, X_2\} \\
\hline
\end{array}
\]

Our earlier WRP design used a straightforward domain enumeration, wherein each binary variable requires two species—one for the value 0 and one for the value 1, marked with a superscript. That is, the assignment \(x_1 = 0\) translates into the injection of species \(X_1^0\), and \(x_1 = 1\) into the injection of species \(X_1^1\). Analogously, the cases \(x_2 = 0\) and \(x_2 = 1\) are represented as \(X_2^0\) and \(X_2^1\), respectively. The concentration of input species is not arbitrary, and must be selected carefully for a specific design. For instance, the input \((x_1, x_2) = (0, 1)\) for the WRP translates to \(X_1^1 = 2\), and \(X_2^0 = 2\).

If we aim to reduce the number of input species and therefore also the number of reactions, then we need to drop the 0/1 representation symmetry. A naive way would be to discard the zero-value species and provide input to a chemical perceptron only if it formally is the value one. The drawback here is that the input pair \((0, 0)\) would be represented as nothing. That is, a chemical perceptron would not know whether and when to produce the output for the input \((0, 0)\). We argue that a true implementation of a chemical logic gate must not treat the zero-valued input as an aberration, even if it is commonly done so in the biochemical computing literature. For instance, chemical systems of the form \(X_1 + X_2 \rightarrow Y\) are commonly said to represent AND gates, because a measurable output \(Y\) is produced only for a simultaneous presence of both reactants, such systems should rather be called two-signal detectors.

The standard solution to deal with the zero-value problem, widely used in digital system design in electrical engineering, is to introduce a clock signal, provided alongside the regular input. Even though this special signal, which we shall denote \(S_{in}\), is strictly speaking, required only for the input \((0, 0)\), we use it for all input cases. The reason is the overall consistency and the functioning of the ASP. Because the goal is to imitate the weight sum \(y = w_0 + w_1 x_1 + w_2 x_2\), we can consider the clock signal \(S_{in}\) the constant-one coefficient (or the constant input \(x_0 = 1\)) of the bias weight \(w_0\), and so each weight has its own input species, and \(S_{in}\) always accompanies the regular input \(X_1\) and \(X_2\) (table 2). This approach simplifies the design and makes the clock signal with the bias processing conceptually independent from the \(X_1\) and \(X_2\) reactions, as we shall discuss in §3.2.
Table 3. Species of (a) the WRP and (b) the ASP. The species are divided into groups according to their purpose and functional characteristics. Note that in addition to the species of the ASP, the TASP also uses an auxiliary output species $Y_{aux}$.

<table>
<thead>
<tr>
<th>group name</th>
<th>species</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>$X, x_i, x_j, x_k, y$</td>
</tr>
<tr>
<td>outputs</td>
<td>$y, y_1$</td>
</tr>
<tr>
<td>weights</td>
<td>$W_0, W_1, W_2$</td>
</tr>
<tr>
<td>total</td>
<td>14</td>
</tr>
<tr>
<td>(b)</td>
<td>$X_1, X_2$</td>
</tr>
<tr>
<td>outputs</td>
<td>$Y$</td>
</tr>
<tr>
<td>weights</td>
<td>$W_0, W_1, W_2$</td>
</tr>
<tr>
<td>total</td>
<td>12</td>
</tr>
</tbody>
</table>

of $Y^1$ is greater than the concentration of $Y^0$, and zero otherwise, i.e. $y = [Y^1] > [Y^0]$. The ASP contains only one output species $Y$; therefore, to distinguish between zero and one output, we impose a threshold concentration $\Theta$, and externally interpret the output as $y = [Y] > \Theta$. Another version of the ASP with internal thresholding distinguishes between two concentration levels: $[Y] = 0$ as the formal zero and $[Y] = 1.5$ as the formal one.

(iii) Weight species
Both the WRP and the ASP use the weight species (table 3), which hold the perceptron’s state and define its functionality by catalysing the input-to-output reactions. Furthermore, there are WRP- and ASP-specific species groups, the purpose of which will be explained later.

3.2. Input-weight integration reactions
To implement the input-weight integration of the two-input perceptron, we must cover the four weight sums from table 1. Because chemically computing the weight sum requires not only addition, but also the more complicated operation of subtraction, we must first address the representation of negative numbers, which is generally problematic in chemistry. Then, we demonstrate how the symmetric and asymmetric approaches to the chemical representation of real numbers affect the design of the WRP and the ASP.

(i) The problem of negative numbers
A concentration is never negative, so we cannot map the values of real variables to the concentrations of associated species directly. Note that the problem of representing the negative numbers is equivalent to the problem of implementing the subtraction operation. Hence, if we restrict the model to positive numbers only, we could only add but not subtract numbers. That might be acceptable for some models, but what if negative numbers cannot be avoided? How can we deal with that in chemistry?

A possible first approach is to introduce a special negative variant of each species and to extend pure addition-based chemical arithmetic with a subtraction operation, wherein the complementary species annihilate when they occur simultaneously in a reactor (figure 1a). This strategy maps each formal real variable $p$ to two species $p^+$ and $p^-$, hence it is an instance of representation symmetry. Intuitively, after the complementary species (whose concentrations we wish to compare or subtract) annihilate, their original state is lost. If the goal is to repeat this comparison, then the system must also maintain back-up copies $\bar{p}^+$ and $\bar{p}^-$ by consuming a fuel. After the comparison is completed, the copies $\bar{p}^+$ and $\bar{p}^-$ are used to restore the original species. Because of the $p \rightarrow \bar{p} \rightarrow p$ reversibility, to prevent an infinite loop, we must precisely time the recovery phase and have a special species catalysing (guarding) the comparison. We implemented this rather cumbersome mechanism for the WLP as described previously [4].

An improved symmetric strategy is to compare the concentrations of species indirectly by their impact on the concurrent reactions they catalyse, and so annihilation occurs at the level of complementary products. Let us consider a chemical system in which a substrate $S$ is transformed to a product $P$ or to $P^+$, depending on the concentrations of two concurrent catalysts $E^+$ and $E^-$, as shown in figure 1b. Assuming the reaction rates of these two reactions are equal, the catalysts represent a positive number if and only if the final concentration of product $[P^+]_{\text{final}} > [P^+]_{\text{init}}$, which holds for $[E^+]_{\text{final}} > [E^+]_{\text{init}}$ otherwise, we represent a negative number. Because all products are derived from substrate $S$, $[P^+]_{\text{final}} + [P^+]_{\text{init}} + [S]_{\text{final}} \leq [S]_{\text{init}}$ and finally $[P^+]_{\text{final}} + [P^+]_{\text{init}} \leq [S]_{\text{init}}$.

What other approach could we take to implementing subtraction in chemistry? The general case may be difficult, but what if a qualitative comparison, rather than precise subtraction, suffices? Because the interpretation of positive and negative numbers is external (performed by us), the mapping from the concentration of catalyst(s) can be arbitrary. To eliminate the representation symmetry, we can keep just one catalyst $E$ and one product $P$, but then all of the substrate $S$ will eventually turn to product $P$. Therefore, we need to reintroduce a competition. Even though a negative catalyst is banned, we can still achieve a race if we introduce a decay of substrate $S \rightarrow \lambda$. Hence, the catalyst $E$ must work against a pressure, which is linear in the concentration of $S$ (figure 1c). The final concentration of product $P$, after the experiment $[P]_{\text{final}}$ depends on the rate of the decay reaction and the concentration of $E$, and is bounded by $[P]_{\text{final}} < [S]_{\text{init}}$. To exploit this mechanism for the representation of real numbers, we set the threshold concentration $\Theta_p$ and $\Theta_s$ such that $[P]_{\text{final}} > \Theta_p$ if and only if $[E]_{\text{final}} > \Theta_s$. For a given initial substrate concentration $[S]_{\text{init}}$, the product concentration threshold $\Theta_p < [S]_{\text{init}}$ and reaction rates, we can determine the threshold concentration of catalyst $\Theta_p$ such that $[E]_{\text{final}} > \Theta_p$ produces the concentration of product $[P]_1$ that is interpreted as positive, or if $[E]_{\text{final}} \leq \Theta_p$ as negative. The relation between the concentration of catalyst $[E]_0$ and the final concentration of product $[P]_{\text{final}}$ and therefore also between the thresholds $\Theta_p$ and $\Theta_s$ is plotted in figure 2a.

An alternative version of the asymmetric comparison forces a catalyst $E$ to compete against annihilation of substrate $S$ and product $P$ (figure 1d). The relation between $[E]_0$ and $[P]_0$ is slightly different (figure 2b), but it again acts as
Figure 1. Implementations of positive and negative numbers in chemistry. (a,b) Symmetric approach using (a) annihilation of complementary species \( P^\oplus \) and \( P^\ominus \) or (b) competition of catalysts \( E^\ominus \) and \( E^\oplus \). (c,d) Asymmetric approach, wherein a single catalyst \( E \) competes with (c) the substrate decay or (d) annihilation of the substrate \( S \) and the product \( P \). \( \lambda \) stands for no species, i.e. in a chemical implementation, an inert waste product. (Online version in colour.)

Figure 2. Relation between the concentration of catalyst \([E]_0\) and the final concentration of product \([P]_{\text{inf}}\), for the asymmetric representation of real numbers by (a) decay of the substrate and (b) the annihilation of substrate and product using different \( k_{\text{cat}} \) rate constants and fixed \( K_m = 0.05 \) of the catalytic \( S \rightarrow P \) reaction (Michaelis–Menten kinetics) using 0.01-step Runge–Kutta4 numerical integration. The rate of the substrate decay as well as the substrate–product annihilation is 1. For a given threshold product concentration \( [E]_0 \), we can determine the associated catalyst threshold \( \Theta_2 \) (x-axis), so all concentrations of catalyst \([E]_0\) to the left of this threshold represent negative numbers, and all concentrations to the right represent positive numbers. The \([P]_{\text{inf}}\) asymptotically reaches the initial concentration of the substrate \([S]_0 = 1\) (upper line) for \([E]_0 \rightarrow \infty\).

Note that the initial concentration of substrate \([S]_0\) restricts the range of representable numbers in both situations, with or without symmetry.

(ii) Weight-race perceptron

The WRP deals with the input-weight integration by letting the weight species compete on the input–output reactions (figure 3). This is an instance of the second symmetric approach (figure 1b), in which the sum of positive and negative numbers is calculated indirectly by the impact of a catalytic species racing on a shared substrate. Whether the WRP produces more \( Y^1 \) or \( Y^0 \) is determined by the cumulative strength of all \( \oplus \) over the \( \ominus \) weights, which includes both the weight concentrations and the catalytic reaction rates. The weight \( W_i^\oplus \) (or \( W_i^\ominus \)) catalyses solely the reaction \( X_i^1 \rightarrow Y^1 \) (or \( X_i^1 \rightarrow Y^0 \)), analogously for \( W_i^\ominus \) and \( W_i^\oplus \). Because the bias weights \( W_0^\ominus \) and \( W_0^\oplus \) are always active, they drive all input-to-output reactions. Note that at this point the formal weight values stop matching the concentrations of the weight species, owing to the nonlinear nature of a catalytic race.

Besides the 13 reactions shown in figure 3, the original definition of the WRP also used a decay of the input species (see table 4a) to guarantee a uniform contribution of weights in the sum, i.e. a fair race. However, it turns out that the WRP can perform well, regardless of the preference among weights, because it can compensate a certain degree of unfairness in the race by non-uniform weight adaptation, and we can always find such a concentration of weight species that the overall production of \( Y^1 \) over \( Y^0 \) will follow a prescribed binary function profile. More importantly, the WRP can perform this process autonomously (that is what we call learning) as presented in

Figure 3. The WRP’s reactions used in the input-weight integration. Nodes represent species, solid lines are reactions, dashed lines are catalysts and \( \lambda \) is no or inert species. The horizontal symmetry is due to the existence of two perceptron’s inputs; the vertical one is the representation symmetry globally spread across the WRP design. (Online version in colour.)
Table 4. The full reaction list for (a) the WRP, (b) the ASP and (c) the extra thresholding reactions of the TASP. Reactions are divided into groups according to common functional characteristics.

<table>
<thead>
<tr>
<th>group</th>
<th>reaction</th>
<th>catalysts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$X^0 \rightarrow Y^1$</td>
<td>$W_0$</td>
</tr>
<tr>
<td>2</td>
<td>$X^0 \rightarrow Y^0$</td>
<td>$W_0$</td>
</tr>
<tr>
<td>3</td>
<td>$X^1 \rightarrow Y^1$</td>
<td>$W_1$</td>
</tr>
<tr>
<td>4</td>
<td>$W_0^1 + W_1^1 \rightarrow \lambda$</td>
<td>$X_1$</td>
</tr>
<tr>
<td>5</td>
<td>$Y^0 \rightarrow Y^1$</td>
<td>$W_2$</td>
</tr>
<tr>
<td>6</td>
<td>$Y^0 \rightarrow \lambda$</td>
<td>$X_2$</td>
</tr>
<tr>
<td>7</td>
<td>$D^0 \rightarrow \lambda$</td>
<td>$X_3$</td>
</tr>
<tr>
<td>8</td>
<td>$D^0 \rightarrow W_0^1$</td>
<td>$Y^1$</td>
</tr>
<tr>
<td>9</td>
<td>$D^0 \rightarrow W_1^1$</td>
<td>$Y^0$</td>
</tr>
<tr>
<td>10</td>
<td>$W_0 + W_1 \rightarrow \lambda$</td>
<td>$X_4$</td>
</tr>
<tr>
<td>11</td>
<td>$W_0^1 \rightarrow W_1$</td>
<td>$X_5$</td>
</tr>
<tr>
<td>12</td>
<td>$Y_{aux} \rightarrow 2Y$</td>
<td>$X_6$</td>
</tr>
<tr>
<td>13</td>
<td>$2Y \rightarrow Y + Y_{aux}$</td>
<td>$X_7$</td>
</tr>
<tr>
<td>14</td>
<td>$Y + Y_{aux} \rightarrow Y_{aux}$</td>
<td>$X_8$</td>
</tr>
<tr>
<td>15</td>
<td>$Y \rightarrow \lambda$</td>
<td>$X_9$</td>
</tr>
</tbody>
</table>

Table 4. (Continued.)

<table>
<thead>
<tr>
<th>group</th>
<th>reaction</th>
<th>catalysts</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>$W_0 + W_1 \rightarrow \lambda$</td>
<td>$X_4$</td>
</tr>
<tr>
<td>11</td>
<td>$W_0^1 \rightarrow W_1$</td>
<td>$X_5$</td>
</tr>
<tr>
<td>12</td>
<td>$Y_{aux} \rightarrow 2Y$</td>
<td>$X_6$</td>
</tr>
<tr>
<td>13</td>
<td>$2Y \rightarrow Y + Y_{aux}$</td>
<td>$X_7$</td>
</tr>
<tr>
<td>14</td>
<td>$Y + Y_{aux} \rightarrow Y_{aux}$</td>
<td>$X_8$</td>
</tr>
<tr>
<td>15</td>
<td>$Y \rightarrow \lambda$</td>
<td>$X_9$</td>
</tr>
</tbody>
</table>

§3.3. The WRP keeps its weight species intact during the input-weight integration, so it is easily reusable, but at the price of having both $\oplus$ and $\ominus$ species variants.

(iii) Asymmetric signal perceptron

In the design of the ASP, we further extend the idea of an unfair race. It exploits the reaction rate setting to ensure that the contribution of weights allows the representation of negative numbers and subtraction, but at the same time avoids the $\oplus$ versus $\ominus$ symmetry of weights. The set of input species in the ASP shrinks from $\{X_1^0, X_1^1, X_2^0, X_2^1\}$ to the three species $\{X_1, X_2, S_{in}\}$ (table 2). The input (clock) signal $S_{in}$ primarily needed for the input pair $(0, 0)$, when neither $X_1$ nor $X_2$ is injected, can be elegantly incorporated to the rest of input pairs to serve an additional purpose. Because the bias weight is always included in the weight sum, regardless of the input (table 1), we can extract the bias processing part and design the ASP such that the signal $S_{in}$ will also be the weight-species $W_0$-specific substrate. Therefore, the input signal $S_{in}$ is always injected and it accompanies the regular input species $X_1$ and $X_2$ (if provided). This is a simpler alternative than hooking the $W_0$ species to all possible inputs as in the WRP. The presence of event signals, the input signal and the penalty signal (introduced in §3.3), is another prominent feature of the ASP.

Now, using the asymmetric representation of numbers by a single catalyst (figure 1c), we obtain three weight species $W_0$, $W_1$ and $W_2$ as opposed to $W_0^1$, $W_1^1$, $W_2^1$, $W_0^0$, $W_1^0$ and $W_2^0$ as required by the WRP. Owing to the introduction of the input signal $S_{in}$ each weight species consumes its own input and adds a portion to the global output species $Y$, as shown in figure 4a. By imposing a certain threshold concentration $\Theta$, we create a system in which each weight species races with its private decay and consequently the concentrations of weights can represent both positive and negative numbers.

On the other hand, this system lacks cross-weight racing because of the disconnection of weight impacts: a small concentration of one weight would never affect the contribution of a different weight to the global output. Because no arrow points out of the species $Y$, once produced $Y$ could not be consumed, therefore the system is additive. The output concentration $[Y]$ would consist of three portions $[Y]_{aux}, [Y]_X$ and $[Y]_X$ corresponding to the output produced from the input species $S_{aux}, X_1$ and $X_2$. Because the weights do not influence one another (their contributions are strictly additive), the output for the
as well and asymptotically reaches the total amount of input when the concentration of weights increases, the output increases which can qualitatively imitate the two-input perceptron. As

\[ \text{bound for this case} \]

\[ \text{output concentration with the same threshold} \]

\[ \text{clock signal} \]

\[ S \]

connected in a cascade may not work properly without extra catalyses. In other words, we need the reaction arrows to races with additive contributions.

What could we do to impose a cross-weight global race? As we mentioned, there is no negative pressure, such as decay, that would interlink the products of positive weight catalyses. In other words, we need the reaction arrows to head not just into, but also out of the output species Y. Naively, introducing a decay of the output species Y → λ would not work, because a negative pressure or consumption must be conditional on the input type. Depending on the presence of \( S \), \( X_1 \) and \( X_2 \), a certain part of negative pressure must be turned on or off. To address that, we replace the original asymmetric building block with a version using an annihilation of the substrate (input) and product, instead of decay with bistability.

\[ \text{What could we do to impose a cross-weight global race?} \]

\[ \text{As we mentioned, there is no negative pressure, such as decay, that would interlink the products of positive weight catalyses. In other words, we need the reaction arrows to head not just into, but also out of the output species Y. Naively, introducing a decay of the output species Y → λ would not work, because a negative pressure or consumption must be conditional on the input type. Depending on the presence of \( S \), \( X_1 \) and \( X_2 \), a certain part of negative pressure must be turned on or off. To address that, we replace the original asymmetric building block with a version using an annihilation of the substrate (input) and product, instead of decay with bistability.} \]

WRP designs are modular. The concentration that corresponds to the formal Boolean output one could match the input concentration if we had a bistable regulator—a mechanism that amplifies the output concentration to specific upper value (representing one) if it exceeds the threshold, or otherwise reduces the output to lower value (representing zero).

Wilhelm [38] proposed the smallest chemical reaction system with bistability using four reactions

\[ S + Y \rightarrow 2X, \]

\[ 2X \rightarrow X + Y, \]

\[ X + Y \rightarrow Y + P \]

and

\[ X \rightarrow P, \]

with two species X and Y, energy source S and inert product P. Because the concentration of S is constant, and P is practically a waste, we can discard them from the reaction set and obtain a system with two species X and Y only. The system has three equilibrium states \( x_1 = 0, y_1 = 0, x_2 = (k_1 - \sqrt{k_1 D})/2k_2, y_2 = x_2^2/k_1, y_2 = (k_1 + \sqrt{k_1 D})/2k_2, y_2 = x_2^2/k_1 \) with discriminant \( D = k_1 - 4k_2k_3 \). The first (lower value) and the third (upper value) solutions are locally stable, the second (threshold value) is unstable, hence if it is perturbed upwards, then the system travels to the upper value; if it is perturbed downwards, then it settles to \((0,0)\) (Figure 5).

We can easily adjust this mechanism for a custom upper value and threshold, and derive a thresholded version of the ASP, the TASP, which requires one extra species \( Y_{aux} \) and four reactions, as shown in table 4c. We calculated the rate constants for the upper value 1.5—the input concentration of the ASP—as \( k_1 = 1, k_2 = 1, k_3 = 0.53 \) and \( k_4 = 0.3 \). Note that besides output Y, its companion species \( Y_{aux} \) is also amplified (upper value 2.25). The threshold for the given rate constants is 0.375 for Y and 0.140625 for \( Y_{aux} \). Note that an active thresholding or a conditional amplification is an additional feature built on the top of the standard ASP (SASP). Other than output interpretation, all features and settings, including input-weight integration and learning, are same. Therefore, the label ASP will refer to both models. If a distinction is needed, then we will use either the standard ASP (SASP) or its thresholded extension (TASP). Unlike the SASP, in which the input species are the sole fuel, the TASP continuously consumes external species, which is kept at constant concentration, to maintain the output at the upper equilibrium.

\[ \text{(v) Execution} \]

We have introduced the WRP and the ASP structurally as a collection of species, reactions and catalysts that model the input-weight integration, which is all it takes to mimic any linearly separable binary function. To illustrate this capability, we execute the WRP and the ASP with the best rate constants found by evolution.

Let us assume that we know the correct concentrations of weight species for a given binary function and, as the first step, we place the weight species molecules into the reactor. (Note that normally we would obtain the weight species concentrations by learning.) Then, we inject one of the input combinations from table 2. The concentration of each input species and the input signal is 2 for the WRP, and 1.5 for the ASP. For instance, the input \((x_1, x_2) = (1, 0)\) is injected.

![Figure 4](http://rsif.royalsocietypublishing.org/)

**Figure 4.** (a) The initial and (b) the final version of the ASP’s reactions used in input-weight integration. Nodes represent species, solid lines are reactions, dashed lines are catalyses and \( \lambda \) is nothing, or an inert species. (Online version in colour.)
to the WRP as \([X_1] = 2\) and \([X_2] = 2\), and as \([S_{\text{in}}] = 1.5\), and \([X_i] = 1.5\) to the ASP. Note that we use concentration as dimensionless quantity, and a wet chemical implementation could scale the initial concentrations to molar or nanomolar unit with the rate constants as needed. Competing weights (catalysts) consume the input species and finally produce the output. Because this process takes some time, we cannot inject input species immediately after the previous pair, but we have to wait until the system settles down. For the WRP, the length of this period is \(S_{\text{WRP}} = 5000\) steps, for the ASP, it is reduced to \(S_{\text{ASP}} = 1000\) steps.

Figure 6 presents a trace of WRP and ASP (both SASP and TASP) execution on four consecutive inputs with the concentration of weight species set according to the NAND function. Thus, going from left to right, we obtain NAND function outputs 1, 1, 1 and 0. The WRP outputs one if, following the annihilation of output species, the species \(Y_1\) remains (solid peaks), otherwise \(Y_0\) (dashed peaks) indicates the binary output zero. For the SASP, the concentration of the single species \(Y\) in terms of its position above or below the threshold determines the output. This is what we call passive thresholding. On the other hand, the TASP actively distinguishes between these two positions and amplifies or diminishes the output accordingly. Unlike the WRP, the output species does not decay in the ASP, hence it must be discarded after each processing.
3.3. Learning and feedback reactions
The input-weight integration part deals with the output production driven by given concentrations of the weights. To alter the predefined weight concentrations, and therefore to alter the predefined functionality, we train the perceptron such that it adheres to the required input–output profile. In this section, we describe two approaches, learning by desired output and reinforcement learning, incarnated in our chemical design.

(i) Weight-race perceptron
Recall that classical supervised learning expects a trainer to feed the perceptron with the binary desired output \( d \). Figure 7b shows a high-level diagram of WRP covering both the input-weight integration as well as learning. By applying the representation symmetry, the desired output \( d \) translates into two species \( D^0 \) and \( D^1 \), and so during each learning step, WRP compares the variant of the actual output \( Y \) against the variant of the desired output \( D \). If \( Y \) matches \( D \), i.e., the species \( Y^0 \) and \( D^0 \), or \( Y^1 \) and \( D^1 \) are simultaneously present in the system, then the output is correct, and the weights remain unaltered (\( D^0 \) or \( D^1 \) disappears by decay). Otherwise, the desired-output species \( D \) transforms to the \( \ominus \) or \( \odot \) version of the weight species \( W \), which are added to (or annihilate with) existing weights. This happens, however, only for those weights that participate in an output production for the current inputs (figure 7a). Now, because the bias weight \( W_0 \) always participates in the output production it gets adapted no matter what input was injected. On the other hand, the weight species \( W_1 \) and \( W_2 \) race on their specific input substrate \( X_1^0 \) and \( X_2^1 \), respectively, therefore, the actual output and input species together catalyse the transformation \( D \rightarrow W_1 \) or \( D \rightarrow W_2 \). The learning rate \( \alpha \) is defined as the concentration of the desired output species, so the more \( D \) we provide, the more the weight concentrations change.

(ii) Asymmetric signal perceptron
As opposed to WRP, the ASP needs just a single output species \( Y \). Here, we interpret the formal binary output by thresholding. As a matter of fact, following the same approach as before and distinguishing, the desired output by two variants \( D^0 \) and \( D^1 \) would be cumbersome. Recall that WRP requires two species—input and actual output—to simultaneously catalyse the transformation of the desired output to the weights. This part of the WRP’s design is rather artificial, because most common reactions have a maximum of one catalyst. The ASP promised fewer and simpler reactions, therefore, we avoid using more than one catalyst simultaneously.

For the ASP, we choose a more biologically plausible alternative for the supervised desired output learning—learning by reinforcement. We introduce a special event signal, the penalty signal \( P \), which represents a reinforcement for an incorrect output (figure 8b). Now, ASP has to decide whether to increase or decrease the concentrations of the weight species \( W_0 \), \( W_1 \), and \( W_2 \). We represent those two options by intermediate species \( W^0 \) and \( W^0 \), and so we let two reactions \( P \rightarrow W^0 \) and \( P \rightarrow W^0 \) compete on the penalty signal \( P \). Following our asymmetric approach to comparison, only one of the reactions, namely \( P \rightarrow W^0 \) has a catalyst \( Y \). The concentration of \( Y \) decides whether weights will be incremented or decremented, and the concentration of \( P \) defines by how much (the learning rate \( \alpha \)). More precisely, because the concentration \( [Y] > 0.5 \) represents one, the presence of the penalty signal \( P \) means we expected ASP to provide zero, therefore the weight concentrations must drop, and so \( P \) should split to more \( W^0 \) than \( W^0 \) molecules. Note that as a consequence of the reinforcement both variants \( W^0 \) and \( W^0 \) are always produced. In addition, compared with the WRP, the ASP does not have to handle the disappearance of the feedback species \( P \), because if the ASP operates as expected, then we skip the injection of \( P \).

Having \( W^0 \) and \( W^0 \), the second step is to decide which weights should be adapted. All reactions used for learning are presented in figure 8b. Similar to the WRP, only the weights responsible for the current output production are altered. Because the bias weight \( W_0 \) is active for the input (clock) signal \( S_w \), which is always present, we can directly draw the reaction from \( W^0 \) to \( W_0 \) and annihilate \( W^0 \) and \( W_0 \) for the weight decrease. By using annihilation, we avoided introducing the intermediate \( W^0 \) species. Again, similar to the WRP, the weights \( W_1 \) and \( W_2 \) are active only for the inputs \( X_1 \) and \( X_2 \), respectively. Thus, we need their substrates from the input-weight integration to catalyse the transformation to the intermediate weight changers \( W^1 \) and \( W^2 \) (which annihilate with weights \( W_1 \) and \( W_2 \)), or the transformation of \( W^0 \) directly to \( W_1 \) and \( W_2 \).

The ASP requires more reactions (10) than WRP (8) to model learning (table 4) mainly because it forbids two simultaneous catalysts per reaction; otherwise, we would have achieved an even larger reduction in the number of reactions.
In addition, we intentionally do not handle the case where the concentrations of the weight-specific concentration changers $W_1$ and $W_2$ exceed the actual concentrations of the weights $W_1$ and $W_2$, respectively. We assume this situation does not occur, thanks to a sufficiently high starting concentration of weights and low concentration of the penalty signal. We satisfy these properties by a genetic search of the rate constants (§4.1).

(iii) Execution

Here, we present an experiment (execution) protocol of the chemical perceptrons to demonstrate their learning capabilities. Note that similar to §3.2, the presented examples have the rate constants set by genetic algorithms.

The initial concentrations of the weight species are drawn from a uniform distribution on the interval $[2,10]$ for the WRP, and the interval $[0.5,1.5]$ for the ASP. The learning rate $\alpha$ is constant throughout the whole training, which translates into the constant concentration of feedback species—the desired output $[D^0] = 2$ or $[D^1] = 2$ for the WRP, and the penalty signal $[P] = 0.2$ for the ASP. The feedback species together with the input species describe the expected input–output behaviour of the perceptrons. After injection of a single input, we need to allow the WRP and the ASP some time to produce the output. In both cases, we wait 100 simulation steps and then automatically inject the desired output (WRP), or first verify the correctness of the output, and then provide a penalty signal or no signal (ASP). This also means that the ASP, unlike the WRP, requires active participation of the trainer or environment. During each learning iteration, we randomly draw one of the four input and feedback combinations, and repeat this process every $S_{\text{WRP}} = 5000$, or $S_{\text{ASP}} = 1000$ steps until a solution is found.

Figure 9 presents a trace of WRP and ASP execution for learning the NAND function, starting from a state where the weight concentrations are set such that they represent the CONST0 setting. Over several learning iterations, the concentrations of the weight species change towards the expected solution. Note that because the ASP does not use decay, we have to manually flush the actual output species $Y$ after each iteration.
4. Results

Here, we present the results of our simulations, covering the learning performance of the WRP, the SASP variant with Michaelis–Menten (SASP MM) and mass-action kinetics (SASP MA), and the TASP with the same two kinetics—TASP MM and TASP MA. Unlike the WPR, the ASP’s reactions do not contain two simultaneous catalysts, so we could directly rewrite catalytic reactions in the mass-action format using standard expansion. In doing so, we ready the ASP for a potential DNA-strand displacement implementation [11,23,24].

4.1. Genetic search

Recall that the WRP and the ASP were introduced as a collection of species and reactions, avoiding the specification of rate constants. Because the space of possible rate constants is large, it would be difficult and time-consuming to sample it in a trial-and-error manner or by exhaustive search. We therefore use a standard genetic algorithm (GA) [39,40] to optimize the rate constants. In effect, our reaction design is a qualitative model, which becomes a quantitative, ODE-driven system once the rate constants are set.

Chromosomes encoding possible solutions are simply vectors of rate constants, which undergo crossover and mutation. The fitness of a chromosome reflects how well a chemical perceptron with the given rate constants (encoded in the chromosome) learns the given binary function. As mentioned in §3.3, during each learning iteration, the chemical perceptron obtains one of the four input and feedback combinations. We repeat this process 120 times sequentially; however, we count only the last 20 learning iterations. The fitness of a single chromosome is then the average over 150 runs for each of the binary functions. For easier accessibility and reproducibility of our results, we included the detailed GA parameter values in the electronic supplementary material, table S3.

We performed 20 evolutionary runs for both SASP variants. In all cases, the fitness quickly climbs above 0.7 and then either settles to a local optimum around 0.75–0.9 or, in 25% of cases, it reaches the maximal values of 0.99–1.0. Because the SASP’s output Y is not produced steadily—it does not act as a constant influx—we could not apply the same rate constants and calculate just the thresholding reactions analytically, therefore, we had to run the GA also for the TASP. Almost all TASP’s evolutionary runs saturated at the maximal fitness, similar to the WRP.

4.2. Learning performance

Because we are interested only in the best-performing instances, we obtained learning performance for the best GA rate constants only (see electronic supplementary material, table S1). We calculated the average learning success rate over 10 000 simulation runs for each of 14 binary functions, where each run consists of 200 training iterations, similar to the fitness evaluation (see §4.1). The results (figure 10) show that the ASP can successfully learn all functions and reaches a nearly perfect final score of 99.5% (SASP MM), 99.3% (SASP MA), 99.999% (TASP MM), and 99.995% (TASP MA). That illustrates our asymmetric design is correct and works properly, even with just a half of the WRP’s number of reactions.

The WRP, which reaches perfect performance of 100%, starts from a balanced distribution where the probabilities of the ⊕ and the ⊖ weight species are equal, and hence initially the output production is evenly split between $Y^0$ and $Y^1$. Furthermore, because of the symmetric design, the learning difficulty of a function and its complement are the same (e.g. OR and NOR). For the SASP, it is challenging to find a balanced initial concentration range. In fact, the GA in all our evolutionary runs drives the rates to the state where the initial probability of output one ($[Y] > 0.5$) and output zero ($[Y] \leq 0.5$) always differs. For the best rate constants, the SASP always starts with a zero output as a CONST0 function (figure 11). Therefore, the SASP has a bias for functions with more zeros in the output, and the learning difficulty of a binary function and its complement do not match in general.

Further, the SASP’s error is very function-specific. For instance, the NAND function has by far the worst performance, i.e. 94.78% (SASP MM), 96.02% (SASP MA). Because the SASP acts initially as CONST0, it must push the output for all, but the last bit above the threshold (figure 9). NAND is also the most difficult function, because it is non-additive, and only the simultaneous presence of $X_1$ and $X_2$ with a low concentration of $W_1$ and $W_2$ annihilates the output.
below the threshold. Besides NAND, only the functions AND, IMPL and CIMPL reach non-perfect scores of 97.5–99.5%. Even though the error is marginal, we speculate it could be eliminated by conducting a more detailed search on the initial weight and input concentrations.

On the other hand, the TASP’s error is negligible (less than 0.005%) and it reaches the expected behaviour faster than the SASP. This is due to a larger gap between the formal output one (1.5) and zero (0), which makes the net effect of the weight-changer reactions responsible for the reinforcement learning driven by the output species qualitatively more distinct. The TASP, compared with the SASP, starts with less unbalanced weight concentrations but still its initial preference of the output zero is 94% for the TASP MM. The TASP MA prefers the output one at 64%. This also implies that the space of possible solutions with different starting bias for the TASP is larger than for the SASP. Table 5 summarizes the features of all our chemical perceptrons. The robustness analysis is presented in the electronic supplementary material, §S1.

5. Conclusion
In this paper, we have significantly reduced the lower bound on the complexity of a chemical system capable of learning. We achieved that by adopting the asymmetric approach to the representation of real numbers in chemistry. Our new model of the ASP is able to mimic a two-input binary perceptron by using 50% fewer reactions than its predecessor. Instead of two complementary species, it determines the output value by either active or passive thresholding on a single species. Furthermore, it can learn all logic functions almost perfectly by a novel chemical implementation of reinforcement learning. It maintains high robustness, although not as high as its predecessor (owing to the absence of structural redundancy).

We showed that the ASP can follow both Michaelis–Menten and mass-action kinetics. The former could be used for potential wet implementations using enzymes, such as deoxyribozymes, the latter could be easily transformed into a DNA-strand displacement circuit. The use of DNA allows one to pick arbitrary sequences to stand for an arbitrary species from an artificial chemistry. In strand displacement systems, populations of these species are typically represented by the populations of single-stranded DNA molecules. These interact with double-stranded gate complexes which mediate transformations between free signals. Soloveichik et al. [23] proved that a strand displacement circuit can approximate, with arbitrarily small error, any artificial chemistry based solely on mass-action kinetics. In a nutshell, the mass-action reaction $X_1 + X_2 \rightarrow X_3$ is translated to three displacement

Table 5. Comparison of the WRP, the SASP and the TASP.

<table>
<thead>
<tr>
<th>attribute</th>
<th>WRP</th>
<th>SASP</th>
<th>TASP</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of species</td>
<td>14</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>number of reactions</td>
<td>30</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>overall learning performance</td>
<td>100%</td>
<td>99.5% (MM)/99.3% (MA)</td>
<td>99.999% (MM)/99.995% (MA)</td>
</tr>
<tr>
<td>rate robustness (50% perturbation)</td>
<td>99.34%</td>
<td>86.8% (MM)/65.62% (MA)</td>
<td>63.65% (MM)/65.76% (MA)</td>
</tr>
<tr>
<td>symmetry of species</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>output representation</td>
<td>$[Y^1] &gt; [Y^0]$</td>
<td>$[Y] &gt; 0.5$</td>
<td>$[Y] = 0$ and $[Y] = 1.5$</td>
</tr>
<tr>
<td>learning feedback</td>
<td>desired output</td>
<td>penalty signal</td>
<td>penalty signal</td>
</tr>
<tr>
<td>time steps per input processing</td>
<td>5000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>ODE numerical approximation</td>
<td>Euler</td>
<td>Runge–Kutta4</td>
<td>Runge–Kutta4</td>
</tr>
</tbody>
</table>

Figure 11. Performance of the SASP MM for each of the 14 linearly separable binary functions averaged over $10^4$ runs. The performance of the mass-action variant is similar and not shown here. (Online version in colour.)
reactions $X_i + L \rightarrow H + B$ (a single strand $X_i$ displaces an upper strand $B$ from the complex $L$), $X_2 + H \rightarrow O + W_1$ (a single strand $X_2$ displaces an upper strand $O$ from the complex $H$), and finally $O + T \rightarrow X_3 + W_2$ (a single strand $O$ displaces an upper strand $X_3$ from the complex $T$), where $L, H, B, O, T, H$ are auxiliary fuel species, and $W_1$ and $W_2$ are waste products. Because the SASP needs just 16 reactions and 12 species, an actual wet implementation is realistic even after the transformation, which would produce 83 strands and 40 displacement reactions. Furthermore, the idea of the asymmetric representation of numbers can be used as a chemical design pattern beyond the implementation of a two-input perceptron.

The TASP with 20 reactions and 13 species could serve as a basic building block for more complicated circuits where several perceptrons cascade in a feed-forward network. A couple of technical details need to be addressed to make this work. The first is the reconstruction of the clock signal $S_{\text{sw}}$, which could be done in a similar way as for the input species, but using a very low threshold value. Second, the propagation of the input to a next compartment hosting another copy of the perceptron is not immediate and depends on the compartment permeability, which determines the shape of the input signal. Last, when a parent compartment is expected to process two or more subcompartments’ output signals, the timing of their simultaneous activation needs to be carefully considered. We do not claim a single perceptron design presented in this paper is universal, and, in fact, we could expect that the input–output setting of a chemical perceptron would vary based on its position in the feed-forward network, i.e. whether it is placed in the input, hidden or output layer. These aspects are beyond the scope of this paper and will be addressed in our future research.

Future work will also focus on exploring whether a similar system could emerge spontaneously rather than by design. We also suggest that our chemical learning system could serve as an interface between functional programming and chemical hardware. That would allow to specify, train and reuse chemical systems without redesigning them. Our chemical learning system may have applications in the area of medical diagnosis and smart medication, where it could replace ‘hard-coded’ solutions.

Acknowledgements. The authors thank Milan N. Stojanovic and Matthew R. Lakin.

Funding statement. This material is based upon work supported by the National Science Foundation under grant no. 1028120/1028238.

References


