Biochemical Threshold Function Implementation with Zero-Order Ultrasensitivity

Wei-Chih Huang, Jie-Hong Jiang, François Fages, Franck Molina

Biochemical Threshold Function Implementation with Zero-Order Ultrasensitivity

Wei-Chih Huang*, Jie-Hong Roland Jiang*, François Fages†, Franck Molina‡
* *Graduate Institute of Electronics Engineering*, *National Taiwan University*, Taipei, Taiwan
† *EP Lifeware*, *Inria Saclay*, Palaiseau, France
‡ *CNRS-Alcediag Sys2diag Lab.*, Monptellier, France

***Abstract*—Engineering biochemical reactions is a key task in synthetic biology to enable biomedical and other applications. The biochemical threshold function is a crucial component in the biosensor circuits to be deployed in living cells or synthetic vesicles for disease diagnosis. In this work, based on the zero-order ultrasensitivity, we propose an economic biochemical implementation of threshold functions with reconfigurable threshold values. We show that the so-constructed threshold function module well approximates the unit step function and allows robust composition with other function modules for complex computation tasks.**

I. Introduction

Synthetic biology is an emerging biotechnology, which is orthogonal to silicon-based nanotechnology and is unique in enabling various applications in, e.g., health, medicine, and life science. An important task in synthetic biology is to engineer biochemical reactions of particular function that takes some molecular species as input and produces some other molecular species as output. For example, in the biosensor circuit design, e.g., for the diagnosis of diabetes [1], biomarkers are taken as input, and the results of syndrome detection are reported through reporter proteins as output. Engineering biochemical reactions for the intended function is one of the primary challenges in this field.

In biosensor circuit design, a threshold function with a configurable threshold value is an important component common and useful in filtering biosignals. In this work, we devise a way to economically implement biochemical threshold function by exploiting the mechanism of zero-order ultrasensitivity [2] (Section II). We further show that the threshold function can be used as a building block to construct general logic and neural network computations, which are widely applicable to biosensor circuit design (Section III). Experimental results demonstrate the feasibility and benefit of our approach (Section IV).

II. Threshold Function Construction

An ideal threshold function is intended to be the unit step function $u_t(x)$, defined by

$$u_t(x) = \begin{cases} 1 & \text{if } x > t \\ 0 & \text{if } x \leq t \end{cases}$$

where t is the called the threshold (value) of the function. In this work, we exploit the mechanism of zero-order ultrasensitivity [2] for threshold function implementation. As to be shown, it allows a good approximation to the unit step function and an easy configuration of the threshold value.

The zero-order ultrasensitivity is achieved by the two reactions:

$$\mathrm{X} \underset{\mathrm{E_X}}{\overset{\mathrm{E_Y}}{\rightleftarrows}} \mathrm{Y}, \tag{1}$$

in the Michaelis-Menten kinetics [3], [4], where a reaction

$$\mathrm{S} \xrightarrow{\mathrm{E}} \mathrm{P} \tag{2}$$

denotes substrate S producing product P via catalyst E. Reaction (2) in the Michaelis-Menten kinetics is a shorthand for the three reactions:

$$\mathrm{S} + \mathrm{E} \underset{\mathrm{k_r}}{\overset{\mathrm{k_f}}{\rightleftarrows}} \mathrm{I} \xrightarrow{\mathrm{k_s}} \mathrm{E} + \mathrm{P} \tag{3}$$

in the mass-action kinetics, where I is the intermediate complex of S and E, and k_f, k_r, k_s are rate constants of their corresponding reactions. Given Reaction (2), a parameter k_S, called the Michaelis constant, is defined by

$$k_S = \frac{k_r + k_s}{k_f},$$

and the reaction rate of Reaction (2) in Michaelis-Menten kinetics can be calculated by

$$\frac{d[\mathrm{P}]}{dt} = \frac{k_s[\mathrm{E}][\mathrm{S}]}{k_S + [\mathrm{S}]}, \tag{4}$$

where the square brackets encompassing a species denote the concentration of the species. Note that the Michaelis-Menten kinetics holds under the assumption

$$[\mathrm{S}] \gg [\mathrm{E}] + k_S, \tag{5}$$

which should be carefully maintained in our later exposition.

To gain insight into Reaction (1), the Goldbeter-Koshland kinetics [5] at a high level has the steady-state solution [6]

$$\frac{[\mathrm{Y}]}{[\mathrm{X}] + [\mathrm{Y}]} = \frac{2v_1 j_2}{b + \sqrt{b^2 - 4(v_2 - v_1)v_1 j_2}}, \tag{6}$$

for $v_1 = k_y[\mathrm{E_Y}]$, $v_2 = k_x[\mathrm{E_X}]$, $j_1 = \frac{k_Y}{[\mathrm{X}]+[\mathrm{Y}]}$, $j_2 = \frac{k_X}{[\mathrm{X}]+[\mathrm{Y}]}$, $b = v_2 - v_1 + j_1 v_2 + j_2 v_1$, where k_y and k_Y are the rate constants of the upper reaction $\mathrm{X} \xrightarrow{\mathrm{E_Y}} \mathrm{Y}$ of Reaction (1), and k_x and k_X are the rate constants of the lower reaction $\mathrm{Y} \xrightarrow{\mathrm{E_X}} \mathrm{X}$ of Reaction (1), each of which correspond to the rate constants k_s and k_S of Reaction (2), respectively. By Equation (6), let $[\mathrm{E_Y}]$ be the input signal and $[\mathrm{Y}]$ be the output signal of the intended threshold function. The threshold value corresponds to the input value of $[\mathrm{E_Y}]$ when $v_1 = v_2$ that maximizes the slope of the tangent of $[\mathrm{Y}]$. Since v_1 is controlled by the input $[\mathrm{E_Y}]$, the threshold value

can be adjusted by setting proper $[\mathrm{E_X}]$. By rewriting Equation (6) to express output $[\mathrm{Y}]$ as a function of input $[\mathrm{E_Y}]$ and letting j_1 and j_2 have small values, we obtain a threshold function, called *Goldbeter-Koshland function*, to approximate the unit step function as shown in Figure 1. The threshold function will be exploited as a building block, as visualized in Figure 2, to achieve complex computation tasks.

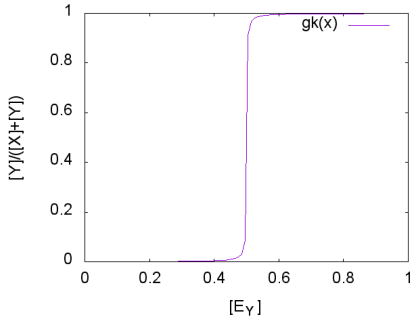


Fig. 1: Plot of Goldbeter-Koshland function, $[\mathrm{X}]+[\mathrm{Y}]=1, k_y=1, v_2=0.5, j_1=j_2=0.001$.

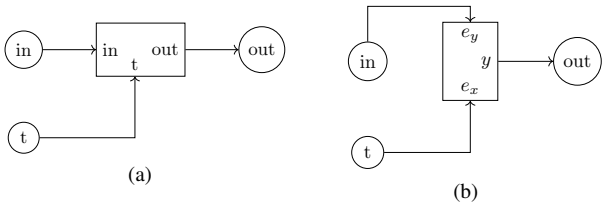


Fig. 2: Block diagram of (a) threshold function; (b) threshold function realized by zero-order ultrasensitivity.

As the Goldbeter-Koshland function is obtained under the Michaelis-Menten assumption of Equation (5), it does not generally work in mass-action kinetics under the six reactions:

$$\begin{aligned} \mathrm{X}+\mathrm{E_Y} &\rightleftarrows \mathrm{I_Y} \longrightarrow \mathrm{E_Y}+\mathrm{Y} \\ \mathrm{X}+\mathrm{E_X} &\longleftarrow \mathrm{I_X} \rightleftarrows \mathrm{E_X}+\mathrm{Y} \end{aligned} \tag{7}$$

According to Equation (5), we need to maintain

$$[\mathrm{Y}] \gg ([\mathrm{E_X}]+k_X), [\mathrm{X}] \gg ([\mathrm{E_Y}]+k_Y). \tag{8}$$

When the threshold function module interacts with other reactions, if the upstream reactions are catalyzed by their inputs and continuously produce $\mathrm{E_Y}$ (recall that $[\mathrm{E_Y}]$ is the input signal of the threshold function), then $[\mathrm{E_Y}]+[\mathrm{I_Y}]$ can gradually increase. Note that, unlike the invariants, $[\mathrm{X}]+[\mathrm{Y}]$, $[\mathrm{E_X}]$ and $[\mathrm{E_Y}]$, in Reaction (1) of Michaelis-Menten kinetics, in Reaction (7) of mass-action kinetics only $[\mathrm{X}]+[\mathrm{I_X}]+[\mathrm{I_Y}]+[\mathrm{Y}]$, $[\mathrm{E_X}]+[\mathrm{I_X}]$ and $[\mathrm{E_Y}]+[\mathrm{I_Y}]$ are invariants. At the equilibrium of Reaction (7), we have

$$[\mathrm{I_Y}]=\frac{[\mathrm{X}]}{k_Y}[\mathrm{E_Y}], \qquad [\mathrm{I_X}]=\frac{[\mathrm{Y}]}{k_X}[\mathrm{E_X}].$$

As mentioned previously, we need small j_1 and j_2, that is, either small k_X and k_Y or large $[\mathrm{X}]+[\mathrm{Y}]$, to make the Goldbeter-Koshland function well approximate the unit step function. The excessive amount of $\mathrm{E_Y}$ and $\mathrm{I_Y}$ makes $[\mathrm{X}]$ and $[\mathrm{Y}]$ small. This breaks the assumption of Equation (8) and makes the output signal $[\mathrm{Y}]$ of the threshold function respond not as intended.

Another issue is that in iterative computation before the next evaluation iteration, the concentration of the input enzyme in Reaction (7) need to be reset by the annihilation reaction:

$$\mathrm{E_Y} \longrightarrow \emptyset\,.$$

Otherwise, $[\mathrm{E_Y}]+[\mathrm{I_Y}]$ may increase after each iteration, and eventually violate the assumption of Equation (8).

The above issues can be resolved by a three-phase (input sampling, input reset, and output valuation) approach in an iteration of computation. The three phases are triggered by a multi-phase oscillator [7].

An alternative better solution, purely combinational, is to add the reaction

$$\mathrm{I_X}+\mathrm{I_Y} \longrightarrow \mathrm{X}+\mathrm{Y}$$

to Reaction (7). This reaction keeps $[\mathrm{E_X}]+[\mathrm{I_X}]$ and $[\mathrm{E_Y}]+[\mathrm{I_Y}]$ invariants without increasing to a large value. Hence the assumption Equation (8) is maintained. This method does not need to reset the concentration of input enzymes in Reaction (7). However, if we want to maintain the concentration of input enzymes, additional identification module in [8] should be used.

III. Threshold Function Application

In this section, we show how to use the threshold function to implement logic and neural network computation.

A. Logic Computation

1) Boolean Operator: For the three Boolean connectives, including inversion $\neg$, conjunction $\wedge$, and disjunction $\vee$, they can be implemented with the threshold function module as shown in Figure 3. Specifically, the inversion operation is achieved by taking $\mathrm{E_X}$ as the input, instead of $\mathrm{E_Y}$; the conjunction operation is achieved by composing a multiplication module [8] in front of the threshold function module; the disjunction operation is achieved by composing an addition module [8] in front of the threshold function module. We set the threshold value of these operations to half of the value representing Boolean true for better noise tolerance.

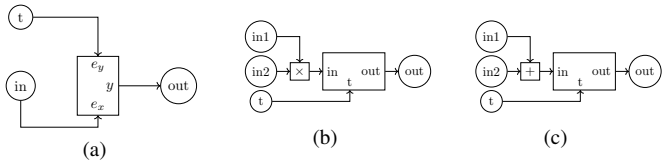


Fig. 3: Block diagram of (a) NOT; (b) AND; (c) OR operator.

2) Comparator: For the three comparison operators "$>$", "$<$", and "$=$", they can be implemented with the threshold function module as shown in Figure 4. Specifically, the "$>$" operator is achieved by the block diagram shown in Figure 4a. For comparison $[\mathrm{A}]>[\mathrm{B}]$, the input B is given as the threshold, and input A is given as the input of the threshold function. An additional input T with small $[\mathrm{T}]$ is added to the threshold value to provide proper resolution of the comparison. (The extra $[\mathrm{T}]$ in the threshold value avoids the output of threshold function fall in the transition region, between 0.4 and 0.6 in x-axis in Figure 1.) The "$<$" operator is achieved by the block diagram shown in Figure 4b. The implementation is the same as the "$>$" operator,

except the inputs A and B are given as the threshold and the input of the threshold function, respectively. The "=" operator is achieved by the block diagram shown in Figure 4c, which can be implemented by the following reactions

$$\emptyset \xrightarrow{\mathrm{A}} \mathrm{A_T}$$
$$\emptyset \xrightarrow{\mathrm{B}} \mathrm{B_T}$$
$$\mathrm{A_T} + \mathrm{B_T} \xrightarrow{\mathrm{k_c}} \emptyset$$
$$\mathrm{A_T} \longrightarrow \mathrm{E_X}$$
$$\mathrm{B_T} \longrightarrow \mathrm{E_X}$$

where k_c is some sufficiently large rate constant. Again an additional input T with small [T] is used as the threshold value $[\mathrm{E_Y}]$ in the threshold function reactions to provide proper resolution of the comparison.

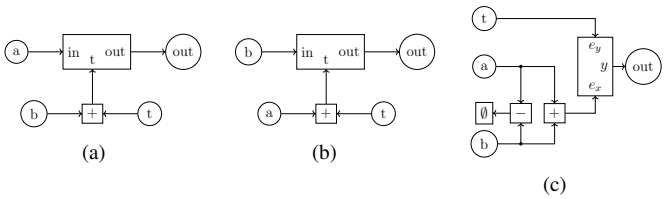


Fig. 4: Block diagram of (a) >; (b) <; (c) = operator.

B. Neural Network Computation

A feedforward artificial neural network consists of layered neurons. Each neuron performs weighted summation of inputs and then feeds the result to an activation function to decide its output. In our neural network implementation, we adopt the reactions for weighted summation computation in [9], but use our new threshold function reactions for the activation function computation.

IV. Experimental Results

We performed experiments on logic and neural network computations discussed in Section III to justify our proposed method. All simulations are done with BIOCHAM [10], and the simulation time unit is in seconds. The threshold function used in these applications is implemented by the purely combinational implementation of zero-order ultrasensitivity reactions in mass-action kinetics, and the corresponding parameters are set as $[\mathrm{X}] + [\mathrm{Y}] = 100$, $k_f = 100$, $k_r = 1$ and $k_s = 10$. The output is re-scaled from $[0 : 100]$ to $[0 : 1]$ for better visualization by using the identification module in [8] with proper catalysis rate constants.

A. Boolean Operators and Comparators

We distinguish Boolean true and false by judging whether the concentration of a species is greater than 0.5 or not. Let A and B be the input species and Z be the output species of the Boolean operators and comparators. The threshold of the Boolean operators is set to 0.5, and the concentration of a species is set to 1 for Boolean true and 0.1 for Boolean false. For the comparators, the extra bias [T] for the "<" and ">" operators is set to 0.2, and k_c and $[\mathrm{E_Y}]$ for the "=" operator is set to 100 and 0.2, respectively. Due to space limitation, only the simulation results of disjunction $\vee$ operator and the "=" comparator are shown, in Figure 5 and Figure 6, respectively.

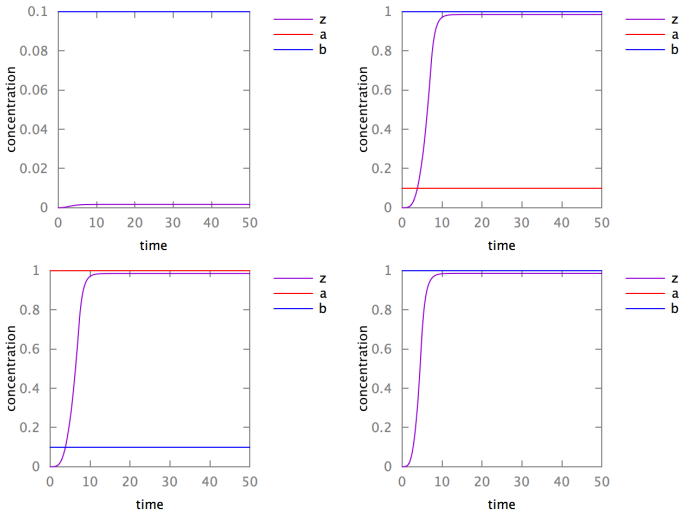


Fig. 5: Simulation results of "$\vee$" operator.

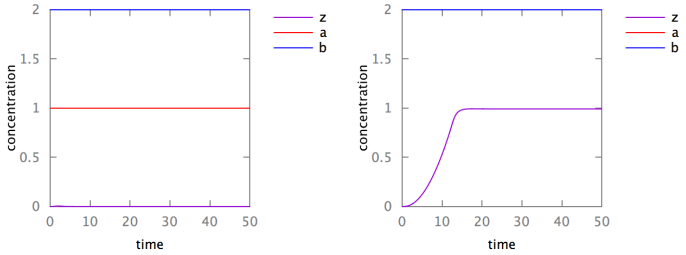


Fig. 6: Simulation results of "=" comparator.

B. Classifier

We implement the classifier in [9], which classifies the input (u_1, u_2) base on the criterion

$$(5u_1 - u_2 \geq 3) \vee [(-u_1 + 2u_2 \geq 1.5) \wedge (u_1 + u_2 \geq 1.5)]$$

The whole network is shown in Figure 7, where the number in the circle indicates the threshold for the neuron, and the number on the line indicates the weight.

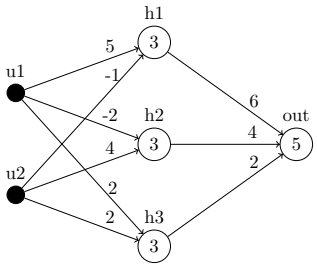
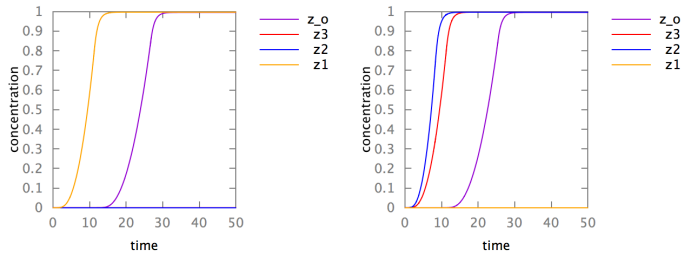


Fig. 7: Neural network classifier.

Sampled simulation results are shown in Figure 8. In the simulation, Z1, Z2, Z3 are the output species of neurons *h1*, *h2*, *h3*, respectively, and $\mathrm{Z_O}$ is the output species of neuron *out*. As can be seen, the classifier works correctly.



(a) $u_1 = 1, u_2 = 0$ (b) $u_1 = 0.5, u_2 = 2$

Fig. 8: Simulation results of classifier.

TABLE I: Results of resource utilization.

	zero-order ultrasensitivity				tanh approximation	
	purely combinational		three-phase			
	#reactions	#species	#reactions	#species	#reaction	#species
$\neg$	13 (0.33)	9 (0.69)	16 (0.41)	11 (0.85)	39 (1.00)	13 (1.00)
$\wedge$	13 (0.32)	10 (0.71)	18 (0.44)	13 (0.93)	41 (1.00)	14 (1.00)
$\vee$	16 (0.38)	11 (0.79)	19 (0.45)	13 (0.93)	42 (1.00)	14 (1.00)
$>$	14 (0.32)	10 (0.67)	17 (0.39)	12 (0.80)	44 (1.00)	15 (1.00)
$<$	14 (0.32)	10 (0.67)	17 (0.39)	12 (0.80)	44 (1.00)	15 (1.00)
$=$	19 (0.42)	13 (0.81)	22 (0.49)	15 (0.94)	45 (1.00)	16 (1.00)
class.	90 (0.52)	72 (0.87)	102 (0.59)	72 (0.87)	173 (1.00)	83 (1.00)

C. Comparison of Different Implementation

To justify that our method is resource-economic, we implemented an alternative threshold function with the tanh approximation of the unit step function defined by

$$f(x) = \frac{s}{2}(1 + \tanh(c \times (x - t))), \tag{9}$$

where parameter s sets the maximum value of the function, c controls the slope of the transition between 0 and s, and t decides the threshold. The corresponding implementation reactions are obtained through the PIVP compilation [11].

Table I compares the resource usage of the proposed method and the alternative method in terms of the numbers of reactions and species used in different computations. The ratios are shown in the parentheses. We note that the purely combinational (resp. three-phase) implementation of zero-order ultrasensitivity requires seven (resp. ten) extra reactions and one (resp. two) extra species for it to work as a module. As can be seen, the reaction counts and species counts in purely combinational implementation of zero-order ultrasensitivity are the fewest and are about 40% and 80%, respectively, of those of the tanh approximation implementation. (In the results of the three-phase approach implementation of zero-order ultrasensitivity in Table I, we exclude the oscillator implementation cost, which uses 18 reactions and 18 species.) As the oscillator can be shared among all threshold function modules, the larger the computation task, the lower the oscillator implementation overhead.

When the computation delay is considered, Figure 9 shows the performance of the three methods on the inversion (NOT) operation. The zero-order ultrasensitivity implementation has slower step transition compared to the tanh implementation. Note that the long response delay of the three-phase approach of zero-order ultrasensitivity implementation is due to the three-phase evaluation. As in biochemical systems, the computation performance is not the major concern. The method based on zero-order ultrasensitivity can be a preferred option.

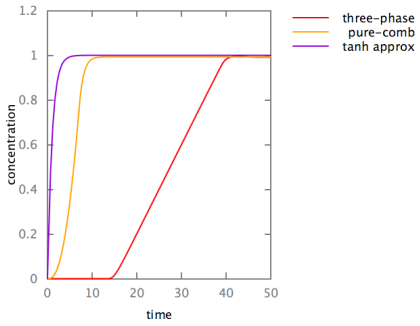


Fig. 9: Comparison of convergence time.

V. Conclusions

We have proposed a new approach to implement the threshold function with configurable threshold value based on the zero-order ultrasensitivity mechanism. The implementation is economic and its reaction pattern can be found in natural reactions, thus amenable for real-world realization. Moreover, we have applied the threshold function as a building block to achieve general logic and neural network computation. Experiments have been conducted to justify the benefit of our method. For future work, we seek biochemical realization of the proposed reactions for wet lab experiments.

Acknowledgments

This work was supported by the BIOPSY project under grant 106-2923-E-002-002-MY3 of the Ministry of Science and Technology (MOST) of Taiwan and the National Research Agency (ANR) of France.

References

[1] A. Courbet, P. Amar, F. Fages, E. Renard, and F. Molina, "Computer-aided biochemical programming of synthetic microreactors as diagnostic devices," *Molecular Systems Biology*, vol. 14, p. e7845, 2018.

[2] J. E. Ferrell Jr. and S. H. Ha, "Ultrasensitivity part I: Michaelian responses and zero-order ultrasensitivity," *Trends in Biochemical Sciences*, vol. 39, no. 10, pp. 496–503, 2014.

[3] K. A. Johnson and R. S. Goody, "The original Michaelis constant: Translation of the 1913 Michaelis-Menten paper," *Biochemistry*, vol. 50, no. 39, pp. 8264–8269, 2011, pMID: 21888353. [Online]. Available: https://doi.org/10.1021/bi201284u

[4] G. E. Briggs and J. B. Haldane, "A note on the kinetics of enzyme action," *The Biochemical Journal*, vol. 19, no. 2, pp. 338–339, 1925. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/16743508

[5] A. Goldbeter and D. E. Koshland Jr., "An amplified sensitivity arising from covalent modification in biological systems," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 78, pp. 6840–4, 1981.

[6] Z. Szallasi, J. Stelling, and V. Periwal, *System Modeling in Cellular Biology: From Concepts to Nuts and Bolts*. The MIT Press, 2006.

[7] M. Vasic, D. Soloveichik, and S. Khurshid, "CRN++: Molecular programming language," in *Int'l Conf. on DNA Computing and Molecular Programming*, 2018.

[8] H. J. Buisman, H. M. M. ten Eikelder, P. A. J. Hilbers, and A. M. L. Liekens, "Computing algebraic functions with biochemical reaction networks," *Artificial Life*, vol. 15, pp. 5–19, 2008.

[9] H. K. Chiang, J.-H. R. Jiang, and F. Fages, "Reconfigurable neuromorphic computation in biochemical systems," in *Int'l Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2015, pp. 937–940.

[10] EP Lifeware. (2017) The biochemical abstract machine BIOCHAM. INRIA Saclay. [Online]. Available: https://lifeware.inria.fr/biocham4/

[11] F. Fages, G. Le Guludec, O. Bournez, and A. Pouly, "Strong Turing completeness of continuous chemical reaction networks and compilation of mixed analog-digital programs," in *Int' Conf. on Computational Methods in Systems Biology (CMSB)*, 2017, pp. 108–127.