Chem

CellPress

Preview

A Language for Molecular Computation

Benjamin L. Foulon,[1] Yuan Liu,[1] Jacob K. Rosenstein,[2] and Brenda M. Rubenstein[1,*]

Advances in molecular computing have been thwarted by the lack of theoretical underpinnings that are both generalizable and experimentally practical. In a recent issue of *iScience*, Dueñas-Díez and Pérez-Mercader begin to fill this void by illustrating the correspondence between Chomsky's hierarchy of formal grammars and example chemical automata.

With semiconductor transistor dimensions approaching fundamental physical limits, researchers have revived an age-old question: are there alternative computing paradigms that would enable high-performance computation at molecular scales? The prospect of using molecules for computation is tantalizing given the staggering amount of information that small quantities of chemicals could encode: if every molecule in a flask could be engineered to compute, Avogadrian ($>10^{23}$) numbers of parallel computations could be performed in just cubic centimeters of volume. Moreover, molecules can undergo collective phase transitions that are inherently nonlinear and possess many degrees of freedom, including vibrational, nuclear, and constitutional states, each of which could be used for information storage and processing on its own intrinsic timescale. Molecular computing may sound far-fetched, but all of biology relies on intricately evolved molecular information processing systems. Despite this promise, many foundational theoretical and experimental questions remain to be resolved before a practical and competitive molecular computer could ever be realized.

Thus far, molecular information demonstrations have overwhelmingly focused on molecular data storage and chemical logic gates. Biomolecular examples of information storage have now become somewhat routine, and the past two decades have seen megabytes of chemical data written and read back using DNA, RNA, metabolites, peptides, and polysaccharides.[1–3] More recently, data have also been successfully written to non-biological small molecules, including Ugi reaction products and phenols.[4,5]

Chemical computation, on the other hand, has been significantly more challenging to realize, because the field still lacks a clear theoretical underpinning for general purpose chemical computations that are experimentally viable (see Chen et al.[6] for the most compelling theory to date). Some of the earliest examples of molecular computing harnessed DNA hybridization to solve optimization problems. In the years since, many efforts have transitioned from designing single-purpose molecules for specific computational demonstrations to designing more universal chemical logic gates that can, in principle, be cascaded to realize more generalizable chemical circuits.[7] At first, many of these gates were designed using biomolecules, but the relatively recent realization that small-molecule chemistries manifest a much wider range of behaviors—and therefore computations—has ushered in the design of small-molecule- and chemical-oscillator-based gates.[8]

Nevertheless, designing chemical logic gates amounts to a conservative approach to chemical computing when compared to the human body, which orchestrates everything from learning tasks to homeostasis not via an assemblage of gates but through more computationally complex feedback loops.[9] It is only in the past few years that researchers have explored approaches that go beyond gates by attempting to map small-molecule reactions to more complex mathematical operations.[4] Without a theoretical framework that can guide the integration of these reactions into meaningful computations, however, many of these efforts amount to shooting in the dark.

In a recent issue of *iScience*, Dueñas-Díez and Pérez-Mercader take a step forward by demonstrating one form of computation, formal language recognition, using non-biological reactions.[10] They examine three chemical systems that can be described by three classes of formal grammars of increasing complexity—regular, context-free, and context-sensitive—from Chomsky's hierarchy. In doing so, they show how reaction network features, such as pH and system-state oscillation, can be employed to perform computational tasks. Their early success points to language-based logic as a potential framework for unifying chemical computation.

[1]Department of Chemistry, Brown University, Providence, RI 02912, USA

[2]School of Engineering, Brown University, Providence, RI 02912, USA

*Correspondence: brenda_rubenstein@brown.edu

https://doi.org/10.1016/j.chempr.2019.11.007

As a first simple example, the authors build a finite automaton (FA) out of a bimolecular reaction and use it to recognize the L_1 regular language. FAs accept or reject patterns based on predefined rules; for elementary bimolecular reactions ($A+B \rightarrow C$), the rule is to produce C if both A and B are present, enabling it to recognize the L_1 language of strings containing at least one A and B. The reaction "accepts" inputs if C is produced and "rejects" them otherwise. The authors use a precipitation reaction, in which C is detectable on sight, to perform the recognition.

To recognize a context-free language, the authors build a 1-stack pushdown automaton (1-PDA) out of a pH reaction network. 1-PDAs are essentially FAs with a stack that can have symbols added (pushed) to and removed (popped) from it. Chemically, this requires (1) a chemical-based stack that can have things added and removed and (2) a way to reject input strings that try to pop symbols from an empty stack. In the authors' reaction network, the pH level acts as a stack, with pH = 7 indicating empty. This network can recognize the Dyck language of well-balanced pairs (e.g., sets of parentheses) represented by additions of weak acid ("open") or strong base ("close"). Adding acid pushes to the stack, whereas adding base pops from the stack. Any string that reduces the pH below 7 implies an imbalance and is immediately rejected. Strings read to their end are accepted only if pH = 7, which implies a balanced number of acid and base additions.

Recognizing more sophisticated context-sensitive languages requires Turing machines (TMs), which can be built from PDAs with multiple interacting stacks. Thus, chemical TMs need multiple measurable quantities that non-linearly depend on each other. In the Belousov-Zhabotinsky (BZ) oscillatory reaction network, there is a non-linear relationship between oscillation frequency (f) and deviations from the maximum redox potential (D). The authors use f as a stack and D as a symbol-processing counter. These quantities depend on the system's levels of oxidizing and reducing agents, along with the basicity; adding to each sequentially corresponds to inputting a string made of symbols a, b, and c. Using this mapping, the network can recognize the L_3 language ($L_3 = (a^n, b^n, c^n)$).

There are several reasons to be encouraged by these early but exciting demonstrations. In contrast with some previous proposals for universal chemical computation, Dueñas-Díez and Pérez-Mercader's is non-biological and yet still practical: each of their automata involves a one-pot reaction and is thus more amenable to future automation. Furthermore, the authors' realization of several language-recognition tasks in different chemistries provides a correspondence between the complexity of a chemical reaction and the difficulty of the language-recognition task. Increasing reaction complexity from a single bimolecular reaction to a set of coupled reactions demonstrates how specific properties of chemical reaction networks are useful in achieving particular computational goals. This provides much-needed intuition to guide the search for chemical reaction networks in the future.

Nonetheless, as the authors concede, this work falls short of establishing a rigorous mapping between chemical reaction networks and automata. So far, this approach seems to offer a new vocabulary to describe existing chemical systems rather than systematic paths for designing future ones. A larger breakthrough would define how to reliably implement a broad family of mathematical forms in chemistry. This will require constraints that allow chemical outputs to be cascaded directly into chemical operators; considerations of yield, speed, and energy; and procedures for determining which sets of formal languages are physically realizable. Here, as in other early work on chemical logic, we must draw an important distinction between observing that a system can be mapped to a Boolean logic gate (for example) versus first defining an arbitrary logic function and then identifying a chemical system that implements it.

Questions also remain about how the proposed networks would scale to larger problems. There may be an Avogadrian number of molecules in any macroscale system, but if they cannot be programmed or read independently, then this parallelism is of limited use. Dueñas-Díez and Pérez-Mercader correctly point out that the complexity of an operation (or formal language) corresponds to its computational power. It is useful to remember that electronic logic gates are trivial in isolation, but they are powerful because they can be reliably interconnected into large networks. Outside of biology, it is still challenging to imagine comparable levels of programmable complexity in chemical networks. This is an immense and worthy challenge, and it is our belief that researchers should not settle for small isolated examples of chemical logic.

One useful measure of a chemical computation could be the degree to which it saves electronic computations. For example, a chemical computation would show a clear benefit if evaluating an input string of length N would have required $O(N^3)$ electronic instructions but only $O(N)$ chemical measurements. In this understandably early example, the simple formal languages that the authors have chosen would require only $O(N)$ instructions to evaluate. Unfortunately, their pH-PDA and BZ systems both still require continuous observation and the same $O(N)$ chemical measurements to classify input sequences as would also be required

electronically, limiting their practical value.

When considering isolated examples, showing that one chemical system can correspond to one particular computation can feel like painting a bullseye around an arrow. Yet, even if it does not signify the end of the tournament, at least it illustrates the rules of the game and provides targets for which to aim. By providing new examples that describe experimental chemical systems using formal languages, Dueñas-Díez and Pérez-Mercader have uncovered a new path that could bring us closer to the elusive target of scalable molecular computation.

ACKNOWLEDGMENTS

This research was supported by funding from the Defense Advanced Research Projects Agency (DARPA W911NF-18-2-0031). The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the US government.

DECLARATION OF INTERESTS

This research relates to patent PCT/US2019/038301: Methods of Chemical Computation.

1. Organick, L., Ang, S.D., Chen, Y.-J., Lopez, R., Yekhanin, S., Makarychev, K., Racz, M.Z., Kamath, G., Gopalan, P., Nguyen, B., et al. (2018). Random access in large-scale DNA data storage. Nat. Biotechnol. *36*, 242–248.

2. Kennedy, E., Arcadia, C.E., Geiser, J., Weber, P.M., Rose, C., Rubenstein, B.M., and Rosenstein, J.K. (2019). Encoding information in synthetic metabolomes. PLoS One *14*, e0217364.

3. Cafferty, B.J., Ten, A.S., Fink, M.J., Morey, S., Preston, D.J., Mrksich, M., and Whitesides, G.M. (2019). Storage of information using small organic molecules. ACS Cent. Sci. *5*, 911–916.

4. Rosenstein, J.K., Rose, C., Reda, S., Weber, P.M., Kim, E., Sello, J., Geiser, J., Kennedy, E., Arcadia, C., Dombroski, A., et al. (2019). Principles of Information Storage in Small-Molecule Mixtures. arXiv, arXiv:1905.02187. https://arxiv.org/abs/1905.02187.

5. Arcadia, C.E., Tann, H., Dombroski, A., Ferguson, K., Chen, S.L., Kim, E., Rose, C., Rubenstein, B.M., Reda, S., and Rosenstein, J.K. (2018). Parallelized Linear Classification with Volumetric Chemical Perceptrons. Proceedings of the IEEE Conference on Rebooting Computing, 1–9.

6. Chen, H.-L., Doty, D., and Soloveichik, D. (2014). Deterministic function computation with chemical reaction networks. Nat. Comput. *13*, 517–534.

7. Seelig, G., Soloveichik, D., Zhang, D.Y., and Winfree, E. (2006). Enzyme-free nucleic acid logic circuits. Science *314*, 1585–1588.

8. de Silva, A.P., and McClenaghan, N.D. (2004). Molecular-scale logic gates. Chemistry *10*, 574–586.

9. Dalchau, N., Szép, G., Hernansaiz-Ballesteros, R., Barnes, C.P., Cardelli, L., Phillips, A., and Csikász-Nagy, A. (2018). Computing with biological switches and clocks. Nat. Comput. *17*, 761–779.

10. Dueñas-Díez, M., and Perez-Mercader, J. (2019). How Chemistry Computes: Language Recognition by Non-Biochemical Chemical Automata. From Finite Automata to Turing Machines. iScience *19*, 514–526.