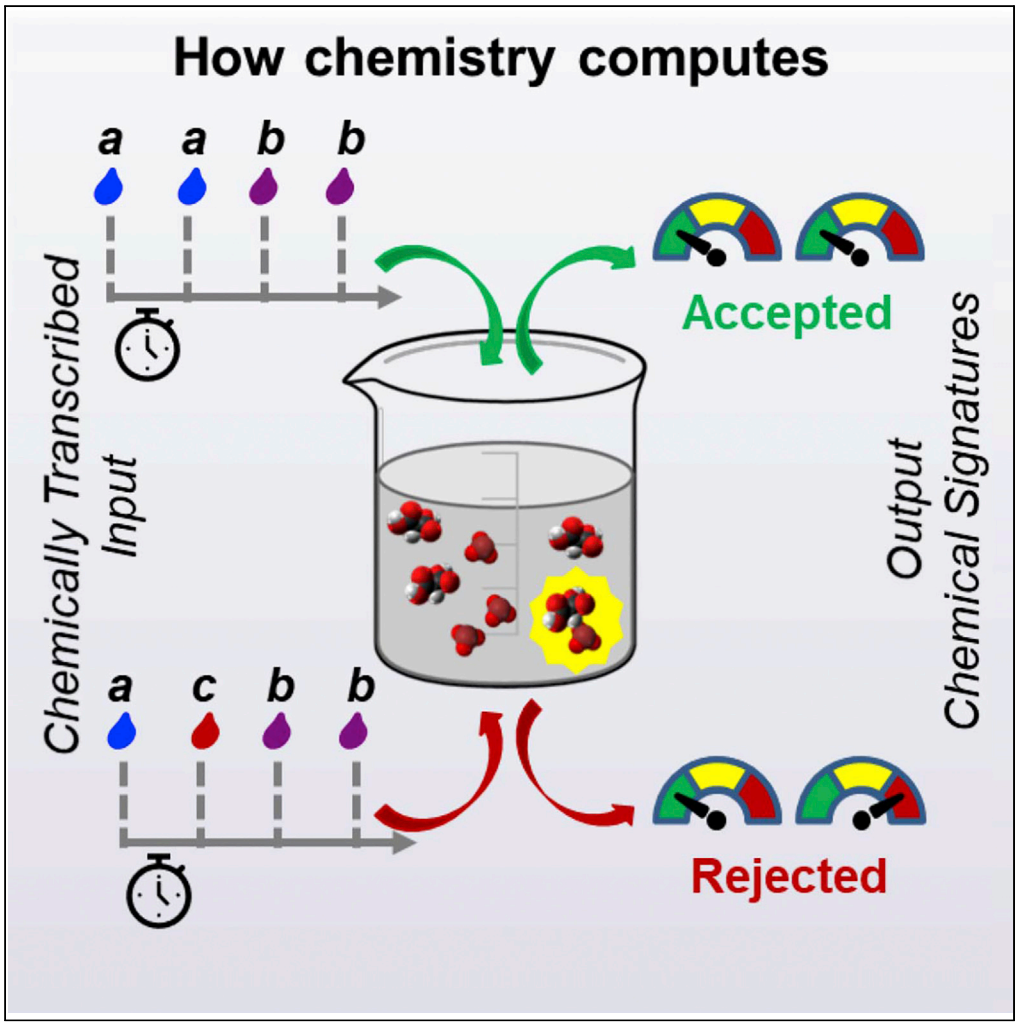
iScience

CellPress

Article

How Chemistry Computes: Language Recognition by Non-Biochemical Chemical Automata. From Finite Automata to Turing Machines



Marta Dueñas-Díez, Juan Pérez-Mercader

jperezmercader@fas.harvard.edu

HIGHLIGHTS

Computations are language recognition events carried out by "computing automata"

Chemical reactions are molecular recognition events equivalent to automata

Words in a language can be represented by sequences of chemical reactants

Inorganic reactions like automata, including Turing machines, recognize languages

Dueñas-Díez & Pérez-Mercader, iScience *19*, 514–526
September 27, 2019
https://doi.org/10.1016/j.isci.2019.08.007

iScience

CellPress

Article

How Chemistry Computes: Language Recognition by Non-Biochemical Chemical Automata. From Finite Automata to Turing Machines

Marta Dueñas-Díez[1,2] and Juan Pérez-Mercader[2,3,4,*]

SUMMARY

Every problem in computing can be cast as decision problems of whether strings are in a language or not. Computations and language recognition are carried out by three classes of automata, the most complex of which is the Turing machine. Living systems compute using biochemistry; in the artificial, computation today is mostly electronic. Thinking of chemical reactions as molecular recognition machines, and without using biochemistry, we realize one automaton in each class by means of one-pot, table top chemical reactors: from the simplest, Finite automata, to the most complex, Turing machines. Language acceptance/rejection criteria by automata can be formulated using energy considerations. Our Turing machine uses the Belousov-Zhabotinsky chemical reaction and checks the same symbol in an Avogadro's number of processors. Our findings have implications for chemical and general computing, artificial intelligence, bioengineering, the study of the origin and presence of life on other planets, and for artificial biology.

INTRODUCTION

Computation is everywhere around us (Moore and Mertens, 2011; Rich, 2008) and is central to life on Earth. Computation takes place not only in the myriad of electronic devices we use daily but also in living systems. In life, biochemistry implements computation via the chemical properties of "organic" matter (Conrad, 1972; Katz, 2012), i.e., using chemical support: inputs are chemical substances, the mechanical processing occurs via chemical reaction mechanisms, and the result is chemical before its transduction into specific functionalities, chemical or otherwise.

More specifically, a computation is (Rich, 2008; Katz, 2012) the process by which information in sequences belonging to a language and consisting of "symbols" in an alphabet are fed to a computing device ("automaton") that recognizes the symbols and is endowed with some rules that allow the automaton to process the symbols according to these rules to eventually deliver an output, such as Acceptance or Rejection of the sequence as belonging or not to the language recognized by the automaton. The pattern of symbols in the sequence is characteristic of the language to which the sequence belongs. We can interpret this process as a metaphor for a chemical reaction or combination of chemical reactions, as one can think of chemical reactions as the result of molecular recognition events that occur precisely, predictably, and repeatably.

Then, we can ask if the power and complexity of biochemistry are necessary to carry out computations using only chemistry. To do so it is useful to recall that languages are classified into an inclusive hierarchy, the Chomsky hierarchy (Rich, 2008; Hopcroft et al., 2007; Chomsky, 1956; Sudkamp, 2006) (cf. Table 1), and that there is a direct correspondence between language complexity and the capabilities of the automata that recognize the language. The most powerful automata are the Turing machines (Turing, 1936).

We answer the aforementioned question by providing non-biochemical realizations of the automata using non-biochemical reactions running in a "one-pot reactor," that is, in a single well-mixed container where multiple rounds of reactions can take place. We do not need any intermediation from either external geometrical aids to channel and direct the chemical fluids or from reactions involving complex biomolecules. To carry out computations we rely fully on the power of molecular recognition associated with the occurrence of chemical reactions and the robustness provided by an Avogadro's number of "processors" working simultaneously. For this, we will introduce in our experimental examples the means for the chemical rendering (translation) of alphabet symbols, the chemical copy of the sequence (transcription), a means

[1]Repsol Technology Lab, c/ Agustín de Betancourt s/n, Móstoles, Madrid 28935, Spain

[2]Department of Earth and Planetary Sciences, Harvard Origins of Life Initiative, Harvard University, 20 Oxford Street, Cambridge, MA 02138, USA

[3]Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA

[4]Lead Contact

*Correspondence: jperezmercader@fas.harvard.edu

https://doi.org/10.1016/j.isci.2019.08.007

Grammars	Languages	Accepting Automata
Type 0 grammars, phase-structure grammars, unrestricted grammars	Recursively enumerable	Turing machine, Non-deterministic Turing machine
Type 1 grammars, context-sensitive grammars	Context-sensitive	Linear-bounded automata (Bounded tape-length Turing machine)
Type 2 grammars, context-free grammars	Context-free	One-Stack Pushdown Automata
Type 3 grammars, regular grammars, left-linear grammars, right-linear grammars	Regular	Deterministic Finite Automata, Non-deterministic Finite Automata

Table 1. The Chomsky Hierarchy

Languages are generated by grammars. By gradually imposing restrictions on them (Hopcroft et al., 2007), grammars are categorized into an inclusive four-level hierarchy, the Chomsky hierarchy. Type-0 are unrestricted grammars and Type-3 the most restricted. Type-1 grammars correspond to Type-1 Languages, which are also called Context Sensitive Languages (CSL). Type-2 and Type-3 correspond to Context Free (CFL) and Regular (RL) languages, respectively. Each class of languages in the Chomsky hierarchy has been characterized as the languages generated by a family of grammars and accepted by a type of machine. The relationships developed between generation and recognition are summarized in this table, which is adapted from p. 338 of Sudkamp (2006). Type 0 and Type 1 grammars are accepted by Turing machines, hence the horizontal line in the Table that separates them from Types 2 and 3.

to feed the sequence (i.e., input the individual symbols), a means to sequentially read the output of the processed sequence, and an autonomous physicochemical indicator to identify the patterns corresponding to sequences in the language recognized by the automaton. After the symbols are fed to the reactor, all the processing and energy used are exclusively chemical. Our purpose is to explore and show how chemical reactions, without using bio-chemistry, reaction-diffusion constructs, or any external auxiliary guidance, do chemical sequence identification in a way equivalent to automata in abstract computations. (We will call this "native chemical computation.") We will not attempt to establish any theory of the correspondence between reaction complexity and automata.

Chemical computing has an interesting and very rich history, making it impossible to offer here a proper review (for more detailed material cf., e.g., Katz, 2012). We will only highlight a few of its many key developments as they relate to the work we present here. In the early 1970s, Conrad (1972) studied how information processing in molecular systems differs from electronic digital computing. The concept of a theoretical chemical diode first suggested by Okamoto et al. (1987) was further developed theoretically by Hjelmfelt et al. (1991) to suggest that neural networks and chemical automata could be constructed connecting several chemical diodes. The Turing completeness of chemical kinetics was theoretically studied by Magnasco (1997). The first experimental realization of chemical AND and OR logic gates using reaction-diffusion was achieved by Tóth and Showalter (1995), followed by XOR gates (Adamatzky and Costello, 2002) and counters (Gorecki et al., 2003). Chemical logic gates are indeed still an active area of research (Gentili et al., 2012; Wang et al., 2016) owing to the difficulties associated with linking many gates to carry out more advanced or meaningful computations such as the ones required to Accept/Reject languages in the Chomsky hierarchy (cf. Table 1). Native strictly chemistry-based computation can be argued (Katz, 2012) to occur in life and has been demonstrated in practice using DNA (Adleman, 1994; Benenson, 2009) or the properties of chromatin (Prohaska et al., 2010; Bryant, 2012). The computational power of DNA has also been studied theoretically (Soloveichik et al., 2010). In summary, most artificial approaches to chemical computing, inspired by living systems, focus on reaction-diffusion systems mostly representing logic gates or use complex biomolecules to solve very specific problems. Finally, an example of a purely chemical, one-pot, non-reaction-diffusion Turing Machine (Turing, 1936) implemented using the Belousov-Zhabotinsky chemical reaction is contained in Pérez-Mercader et al. (2017).

In what follows we demonstrate native "one-pot-reactor" experimental realizations of computations for the recognition of well-known languages at different levels in the Chomsky hierarchy and provide a comparison

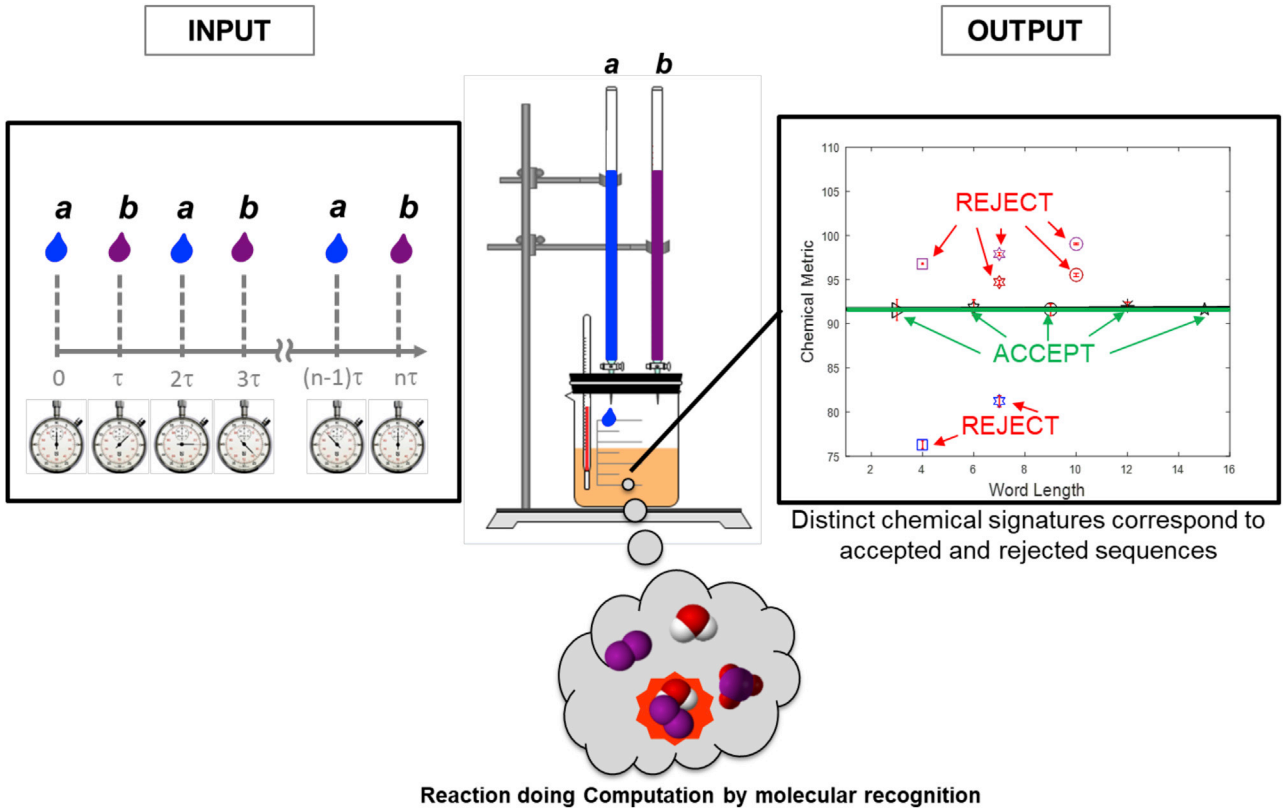


Figure 1. Schematics of Language Recognition by a Chemical Automaton

The chemical automaton comprises three subsystems: (1) a well-mixed reactor where computation takes place; (2) an input scheduler that feeds sequentially the translation into chemical aliquots of the input word letters, one symbol at a time with an individual processing time τ; and (3) a monitoring system to record and follow the automaton's response as a chemical metric. The chemical automaton produces distinct chemical signatures for accepted and rejected inputs (cf. Figures 2 and 4). Just as its abstract counterpart, the chemical machine can reject strings during computation or at the end of computation.

of the corresponding chemical and abstract automata. We also present a suitable physico-chemical metric to characterize the result of the automaton's computation.

RESULTS AND DISCUSSION

Methodology Used to Implement One-Pot Native Chemical Computation

The input to be computed is represented by a sequence of symbols from a chemical alphabet in which each letter corresponds (is translated) to a certain constant amount, or aliquot, of a carefully chosen reacting chemical species. The input is added aliquot after aliquot to a one-pot reactor at constant time intervals (Pérez-Mercader et al., 2017).

The processing of each letter (which is exclusively carried out by chemistry) consists of its chemical species selectively activating specific pathways (Dueñas-Díez and Pérez-Mercader, 2019) in a chemical reaction mechanism and, correspondingly, altering the extent of reaction in a systematic way. Every symbol is allotted a common time interval for processing before adding the next symbol. This interval was experimentally selected on the basis of transient dissipation. Finally, the output of the computation is in the form of a distinct chemical response; that is, for a given automata/language combination, the chemical behavior associated with rejected sequences is different from the chemical behavior associated to accepted sequences or "words in the language of the automaton" (Pérez-Mercader et al., 2017) (cf. Figure 1). The Result of the computation are reaction-generated signatures that do (Accept) or do not (Reject) match a predetermined language and automaton-dependent physico-chemical pattern. Naturally, we expect that such distinct chemical responses correspond to some distinct thermodynamic signatures as well as pathways (Dueñas-Díez and Pérez-Mercader, 2019).

Our methodology is the following. First, we choose a specific and well-known language from a level in the Chomsky hierarchy to be recognized by the chemical automaton. We then identify the class of abstract automaton needed to recognize the language. We translate this into specific requirements for the automaton's chemical reaction mechanism and the nature of the involved chemical species (whether they are reactants, products or intermediates). In turn, the chemical requirements guide us in the appropriate selection of chemicals to represent the symbols in the alphabet. Then, the specific quantitative recipes for the initial conditions and alphabet aliquots are selected taking into account practical experimental realization considerations (for example, so that the chemical reaction monitoring system allows the unimpeded detection of automaton responses). Finally, a comprehensive set of sequences, some belonging to the language accepted by the automaton and others not belonging to the language are tested experimentally.

Automata Hierarchy

As already mentioned, input of information for a computation involves (Rich, 2008; Hopcroft et al., 2007; Brookshear, 1989; Cohen, 1991; Linz, 2012) sequences of symbols (strings) expressing said information in some formal language L using an alphabet Σ. The strings are fed to an automaton that Rejects or Accepts (recognizes) them as words in that language. This result can then be used for subsequent actions or functions. Languages are generated by grammars and they can be classified in a multi-level inclusive hierarchy, the Chomsky hierarchy (cf. Table 1).

Computing automata (Rich, 2008; Hopcroft et al., 2007; Cohen, 1991) consist of at least one "Finite Automaton" (FA), one or more "tapes" and can have one or more "stacks." The FA is a machine responding to symbols in Σ, assumes a finite number of predetermined states, and produces some specific response depending on tape-supplied input. The tape is a construct through which a problem-sequence of symbols is sequentially fed to the FA. The tape can have specific symbols, such as "#", to indicate beginning and end of sequence. When required by the nature of L, an FA can be complemented with a "stack" (Oettinger, 1961): another construct endowed with additional rules enabling temporary storage (memory) and retrieval of information during computation.

In correspondence with the languages they accept and their place in the Chomsky hierarchy, automata themselves can be classified into a hierarchy (Rich, 2008; Hopcroft et al., 2007; Brookshear, 1989; Cohen, 1991; Linz, 2012), which is shown in Table 1. The simplest, Finite Automata (FA), recognize Regular Languages. Next up in the hierarchy are one-stack Push Down Automata (1-PDA) recognizing Context Free Languages. At the top of this hierarchy are Turing Machines (equivalent to a PDA with two or more stacks [Minsky, 1961; Minsky, 1967]) which recognize Context Sensitive, Recursive, and Recursive Enumerable Languages.

A Turing machine (TM) is an FA with a finite but potentially unbounded read-write tape and head (Harrison, 1978). Because in general it uses an unbounded tape the TM is a theoretical construction. However, in any actual physical realization of a TM the tape is necessarily bounded (Minsky, 1967) and can, therefore, be built from two stacks (Cohen, 1991; Searls, 2012; Floyd and Beigel, 1994). Such Turing Machines are called Linear Bounded Automata (Harrison, 1978; Searls, 2012) and accept Context Sensitive Languages. (Note that if the tape in this class of Turing machine is limited to n squares of tape then, by suitably enlarging its alphabet to k-tuples, the automata can increase its effective tape size to $k \times n$ for any fixed k. Hence the word "linear" in its name [Harrison, 1978].)

Next we provide detailed "one-pot-reactor" experimental realizations for one automaton in each of the above-mentioned automata classes capable of recognizing well-known standard languages in the hierarchy.

Chemical Finite Automaton

Chemistry easily carries out an FA computation. Any elementary bimolecular reaction in an aqueous medium

$$A + B \rightarrow C$$

recognizes a regular language (RL), the simplest languages (Hopcroft et al., 2007; Cohen, 1991). In fact, this reaction recognizes the regular language L_1 of all words with a pattern of at least one a and at least

Figure 2. Experimental Recognition of Words in L_1 and L_2 by a Chemical Finite Automaton and a Chemical One-Stack Pushdown Automaton, Respectively

In our realization of an FA (cf. Figure S1), the chemical signature for an accepted word in L_1 is the formation of a white $AgIO_3$ precipitate: sequence "aab" is accepted (A, top), whereas sequence "aaa" (A, bottom) is rejected. The chemical signature for an accepted word in L_2 by our realization of a one-stack PDA is that the pH lies above or at the midpoint during the computation and exactly at the midpoint at the end of computation: sequences ()()(), (())(), and ((())) are all accepted (see plots Ba to Bc), whereas sequences)()()(and ())()) are rejected upon reading the first and third symbol, respectively, as the pH falls below the midpoint value (see plots Bd and Be). Sequence ()()((is rejected at the end of computation since the final pH is above the midpoint (Bf). By using an appropriate pH indicator, the chemical computation can be followed also by observing the color.

one *b* (Hopcroft et al., 2007; Cohen, 1991). (Note this does not involve counting or memory.) Before the computation starts, the automaton, a continuously stirred reactor, contains only deionized water (cf. Figure 2). The input string can be chemically transcribed by rendering (translating) the *a*'s and *b*'s as aliquots of species A and B. An input string (e.g., "aab", "baa", "ab", "aaabbbb") is sequentially fed to the reactor from the left, at regular time intervals much longer than the (common) symbol-feed time. The chemical state of the reactor changes accordingly. Words in L_1 will generate chemical C. Thus, the presence of C in the reactor indicates that the machine has Accepted the string and becomes the result of this computation.

A chemical realization of this FA is provided, for example, by the precipitation reaction:

$$KIO_3 + AgNO_3 \rightarrow AgIO_3\,(s) + KNO_3.$$

If during the computation a white precipitate of silver iodate is observed, the input string has been Accepted, whereas if at the end of the string the solution is free from precipitate the string has been Rejected (cf. Figure 2A, Transparent Methods Section 1 and Figure S1).

Alternatively, since we do our experiments at constant pressure and temperature, we could look at the enthalpy yield (Dueñas-Díez and Pérez-Mercader, 2019) $Y^{(FA)}_{Enthalpy\ End}(\%)$ defined as the ratio of reaction (precipitation) enthalpy at the end of computation to the total enthalpy in the reactants. As expected, we find that any word Not-Accepted by the FA gives $Y^{(FA)}_{Enthalpy\ End}(\%) = 0$, whereas for Accepted words $Y^{(FA)}_{Enthalpy\ End}(\%) \neq 0$ (Dueñas-Díez and Pérez-Mercader, 2019). (Note also that an ON-OFF switch is an FA [Hopcroft et al., 2007] that can be appropriately represented by a straightforward combination of simple reactions.)

Chemical One-Stack Push-down Automaton

Next in the language hierarchy are Context Free Languages (CFL) (Bar-Hillel et al., 1961). These require counting and are recognized by one-stack PDAs: essentially an FA provided with one "stack." The stack can be used to store and read a string of arbitrary length but can only be modified at its top (think of the stack as a cafeteria tray dispenser [Oettinger, 1961]). Operation "push" adds a new symbol to the top of the stack, and operation "pop" removes a symbol from the top of the stack (Hopcroft et al., 2007; Cohen, 1991). By analogy with its computer science counterpart the chemical realization of a stack requires chemical intermediates acting as the product of one reaction ("push") and as reactant for other reactions ("pop"). We therefore need a system of multiple reactions that are interconnected through common intermediate chemical species.

A typical CFL is the language L_2 of well-balanced bracket-pairs (or *Dyck words* [Weisstein, 2009]): strings of open "(" and closed ")" parentheses are well-balanced if during computation the number of ")" never exceeds the number of "(", and, at the end of computation the number of "(" equals the number of ")". The sequences ()()(), ((())), and (()()) are examples of Dyck words.

For a chemical realization of a one-stack PDA parenthesis checker we can use a *pH*-reaction network (cf. Transparent Methods Section 2). We transcribe the input string so that each "(" is an aliquot of a strong base, such as Sodium Hydroxide, and the ")" is an aliquot of a weak (diprotic) acid, for example, Malonic Acid, with volume selected to neutralize one aliquot of base to the first midpoint (Petrucci et al., 2011). Finally, the beginning and end of expression symbol "#" will be an aliquot of a *pH* indicator, e.g., Methyl Red for the acid-base system used below (which changes color around the first midpoint).

At the beginning of a computation, the *pH*-PDA reaction vessel contains deionized water and an aliquot of the *pH* indicator. As before, an input string is then fed sequentially. This leads to changes in *pH* whose value we assign to the stack. The *pH*-PDA will Accept the input string if during the computation the $pH \geq$ Midpoint-*pH* but is at Midpoint-*pH* (empty stack) at the end of computation, i.e., after adding "#". Conversely, the *pH*-PDA Rejects an input string if the *pH* falls below the Midpoint-*pH* at any stage during computation (excess of ")", and attempting to "pop" from an already empty stack) or if the *pH* is larger than the Midpoint-*pH* at the end of computation (excess of "(", or the stack is "not empty") (cf. Figure 2B).

The Result can again be inferred in two different ways, (1) by tracking the evolution of *pH* as the problem sequence is processed and then comparing this with the midpoint *pH* value at maximum reaction extent for this automaton/recipe combination or, equivalently, (2) by looking at the enthalpy yield (Dueñas-Díez and Pérez-Mercader, 2019) of the problem sequence. In the latter case we discover that $Y^{(1-PDA)}_{Enthalpy\ End}(\%)$ is maximum (Dueñas-Díez and Pérez-Mercader, 2019) and independent of word length for every word accepted by the one-stack PDA.

Chemical Turing Machine (I). Formulation

At the top levels of the language hierarchy are the most complex languages requiring increasingly flexible memory and grammars (Cohen, 1991): Context-Sensitive, Recursive, and Recursive Enumerable Languages (REL). These are recognized by Turing machines, a class of automata containing automata equivalent to a two or more stacks PDA (Hopcroft et al., 2007; Cohen, 1991; Minsky, 1961, 1967), including the Linear Bounded Automata (Linz, 2012; Harrison, 1978) (LBA), which, as already mentioned, are a class of Turing machines with predictable memory requirement: the computation uses exclusively the tape cells occupied by the input, and not an unbounded length tape, a clear restriction suitable for physical machines as in the case of our experimental realizations (Minsky, 1967). The two stacks must be in the same automaton and interact. To generalize from a one- to a two-stacks chemical PDA we can chose a chemical system with at least two interdependent chemical observables. The necessity to handle non-linearity in the

computation (Lloyd, 1992) together with the interrelation among reaction observables point us in the direction of non-linear chemistry and, for example, redox oscillators (Tyson, 1985; Field et al., 1972). During the computation, the instantaneous oscillation frequency, *f*, and the difference *D* between the mean of redox potential in each oscillation and its maximum redox potential, will each be in one of two stacks. The non-linear chemical kinetics manifests as a non-linear relationship (*f(D)*) between *f* and *D* (cf. Figure 4A).

A classical textbook example of a Context Sensitive language recognized by Turing machines (Hopcroft et al., 2007; Cohen, 1991) in the Linear Bounded Automaton class, but not by any automaton at the lower levels of the hierarchy, is the language $L_3 = \{a^n b^n c^n$, where $n \geq 1\}$ over the three-symbol alphabet $\Sigma = \{a,b,c\}$. This is the language of all strings consisting of *n a*'s followed by *n b*'s and *n c*'s; for example, "*abc*", "*aabbcc*", "*aaabbbccc*", "*aaaabbbbcccc*". We chose L_3 as the language for our experimental implementation of a chemical Turing machine.

To implement a chemical Turing machine (Pérez-Mercader et al., 2017) we can use the non-linear oscillatory chemistry in the Belousov-Zhabotinsky (BZ) reaction network, whose major overall process when using malonic acid and sodium bromate (Field et al., 1972) is

$$3BrO_3^- + 5CH_2(COOH)_2 + 3H^+ \rightarrow 3BrCH(COOH)_2 + 4CO_2 + 2HCOOH + 5H_2O.$$

Discovered by Belousov in the early 1950s (Belousov, 1959; Winfree, 1984) when he was trying to find a non-biochemical analog for glycolysis (Ball, 1999), the BZ reaction is arguably the best studied chemical oscillator. Although any other BZ recipe can be used, we will use a standard recipe with ruthenium as catalyst and sodium bromate and malonic acid (MA) as oxidizing and reducing reactants, respectively (cf. Transparent Methods Section 3), all in the presence of an acidic environment provided, in our recipes, by sulfuric acid.

A key aspect in the design and implementation of chemical automata is the selection and assignment of chemicals representing the input alphabet and machine symbols (cf. Transparent Methods Sections 1, 2 and 3). A chemical is adequately assigned if it predominantly enhances a pathway within the reaction mechanism that maps to a distinct, repeatable, and systematic (observable) chemical response. Based on extensive computer simulations (cf. Transparent Methods Section 4) and experimental testing of the BZ system, we identified aliquots of the BZ reactants A (sodium bromate) and B (MA) with symbols *a* and *b* in the language, respectively (cf. Supplemental Information section 4). For *c*, we need a substance acting on the two interrelated stacks differently from A and B. In fact, the reaction's *pH*-value provides room for symbol *c*, which we can render chemically (translate) by using an aliquot of sodium hydroxide C. We can also identify an aliquot of the catalyst (Ru-tris(bpy)) as the begin-end of sequence tape-symbol #. This assignment of alphabet symbols provides a mapping to distinct dominant pathways in the well-known Field-Körös-Noyes (FKN) kinetic mechanism (Field et al., 1972) of BZ (cf. Transparent Methods Section 5, and Figure 3) and leads to distinct dynamical behaviors for the observed redox potential. Finally, each symbol will be processed in the reactor during a common time interval, τ, chosen in our experiments to be 7.5 min, which we found provides ample time for fast reaction transients to dissipate.

When the computation starts, the BZ-TM reactor contains only deionized water and a strong acid (sulfuric acid in our recipes). Beginning with "#", and as the chemical input string is sequentially fed to the reactor, the amplitude and frequency of oscillations vary systematically in a manner (the frequency for each group of oscillations behaves [Tyson, 1985] as $f \sim [BrO_3^-]^{\alpha} \times [MA]^{\beta} \times [NaOH]^{-\gamma}$) whereby $\alpha > \beta \sim \gamma > 0$ (cf. Supplemental Information section 6) that depends on the nature and order of the symbols we feed (that is, on the order of the aliquot sequence of *a*, *b*, *c*, or #). During computation, the instantaneous oscillation frequency of the redox oscillator, *f*, and the difference *D* between the mean of redox potential in each oscillation and its maximum redox potential (cf. Transparent Methods Section 3), will each "be" in one of two stacks. The non-linear chemical kinetics manifests as a non-linear relationship, *f(D)*, between *f* and *D*. We denote the value of *f(D)* after the processing of the end-of-word symbol # (Hopcroft et al., 2007), by $f_{\#} = f(D_{\#})$. The more *a*'s the higher the initial frequency upon onset of oscillations; then, as *b*'s are added, the frequency increases (that is, an element is "pushed" on the stack) and finally, as *c*'s are added, the frequency decreases (an element is "popped" from the stack) (cf. Figure 4 and Video S1). We also see (cf. Figure 4 and Transparent Methods Section 7) that *D* functions as a counter for the total number of symbols that have been processed so far.

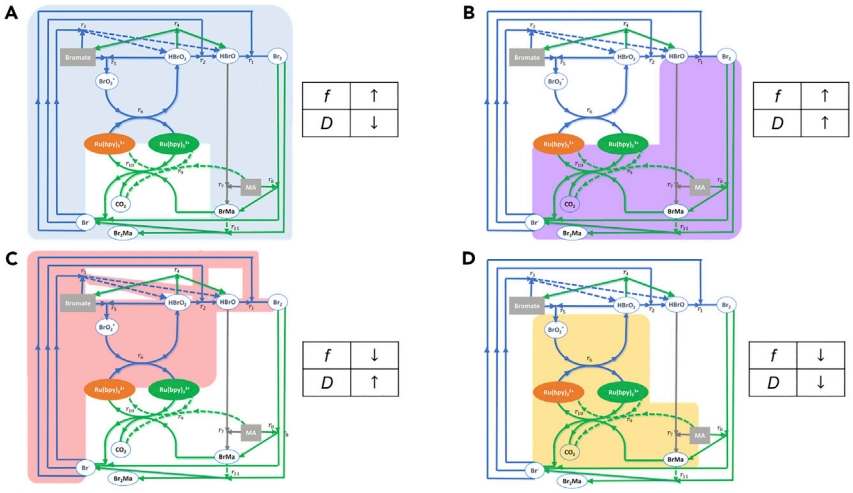


Figure 3. Effect of Symbol Aliquots on Dominantly Affected Pathways in the Extended FKN Model of Belousov-Zhabotinsky

In the FKN mechanism shown in the four panels, reactions consuming H^+ are shown in blue, reactions producing H^+ are shown in green, and reactions that are pH independent are in gray. The colored background areas indicate the predominantly enhanced or diminished pathways by the aliquot symbols, and the tables show the effect on the two key descriptors of the oscillations, f and D. (A) corresponds to bromate aliquots where the light blue pathway is predominantly enhanced; (B) corresponds to malonic acid (MA) aliquots where the purple pathway is predominantly enhanced; (C) corresponds to NaOH aliquots where the red pathways is predominantly diminished; and (D) corresponds to the catalyst aliquots where the orange pathway is enhanced (c.f. Transparent Methods sections 4 and 5).

Chemical Turing Machine (II): Sequence-Word Acceptance or Rejection

With each symbol fed into the reactor, the BZ reaction network is driven through symbol-specific (as already emphasized sequence-dependent) pathways (refs. Dueñas-Díez and Pérez-Mercader [2019] and Transparent Methods Section 5). Thus, as the letter sequence is processed, the reactor produces a specific recipe-dependent and sequence-dependent profile of concentrations of products and intermediates. This affects the extent of reaction and manifests as specific BZ-oscillation frequency/amplitude combinations (cf. Figures 4, S3, and S4 and Video S1). That is, the extent of reaction depends on the order of aliquots in the sequence being fed to the TM reactor, i.e., on the particular word being processed.

The various "out-of-order" signatures for words not in L_3 can be explained as follows. Each symbol has an associated distinct pathway in the reaction network. Hence, if the aliquot added is for the same symbol as the previous one, the pathway is not changed but reinforced. In contrast, when the aliquot is different, the reaction is shifted from one dominant pathway to another pathway, thus reconfiguring the key intermediate concentrations and, in turn, leading to distinctive changes in the oscillatory patterns. The change from one pathway, say 1, to say pathway 2 impacts the oscillations differently than going from pathway 2 to pathway 1. This is what allows the machine to give unique distinctive behaviors for out-of-order substrings (as discussed and shown in Transparent Methods Section 7).

For sequences with symbol order not in L_3 (e.g., "bca", "abc", "abccb", …) our L_3-specific chemical TM rejects them as soon as the letter in the wrong order is detected by the reaction because they generate distinct dynamic oscillatory profiles, see cf. Figure S3 and Transparent Methods Section 7, which are consistent with the abstract L_3-specific LBA-TM counterpart (Cohen, 1991). The sequences tested in our experiments not only covered all distinct scenarios leading to a reject, but we also tested "worst-case" sequences, i.e., sequences not recognized but nearest, both abstractly and chemically, to accepted words.

To select the recipe for symbol aliquots we may require, for example, that considered as a function of word length the production in our semi-batch TM-reactor of the final product of the overall BZ reaction, CO_2, be minimal (a useful requirement for our experimental realization since produced gas interferes with accurate redox potential monitoring for long n's). The parabolic curve in Figure 4 displays how for a recipe so selected the two stacks are non-linearly interrelated after the end-of- sequence symbol is processed for

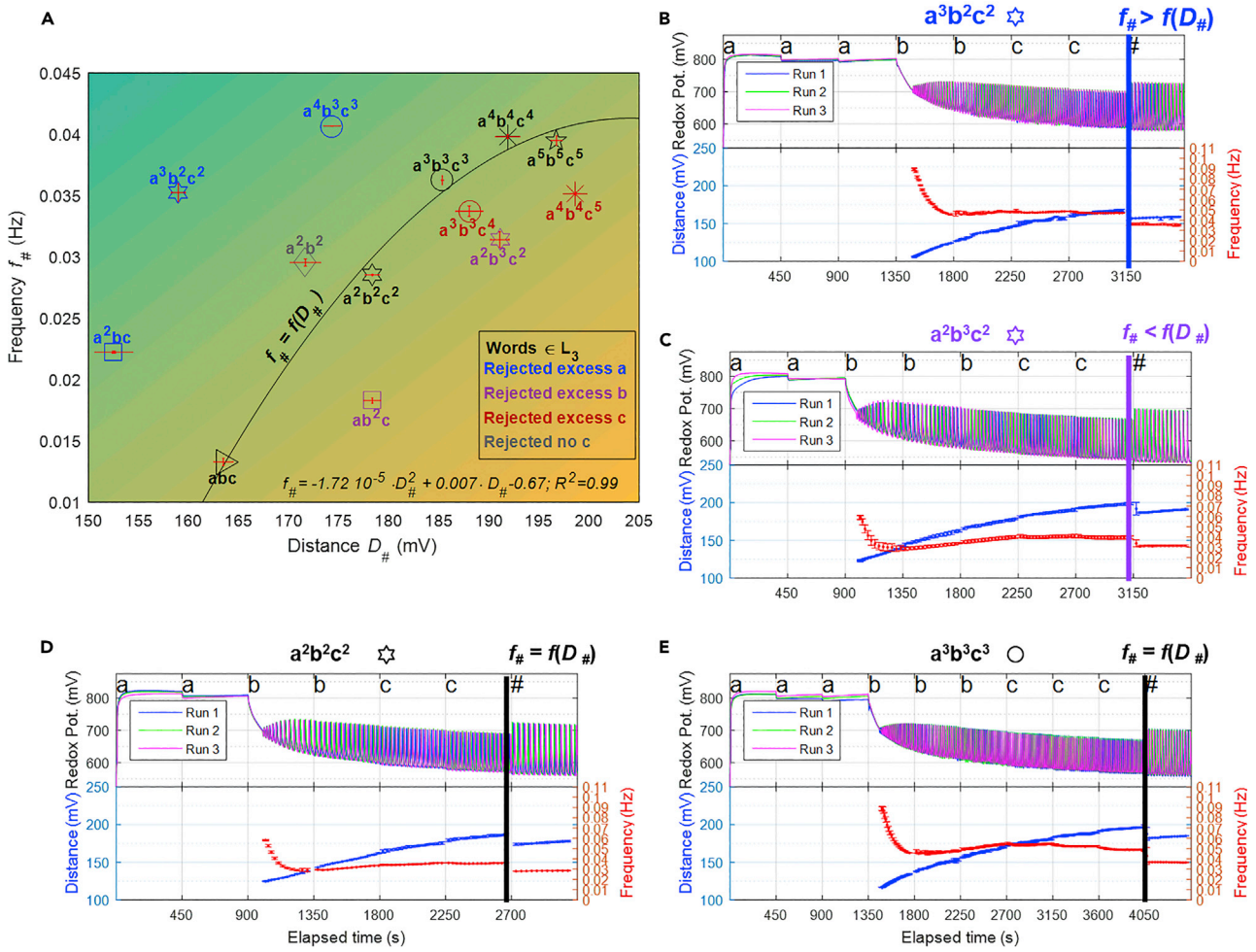
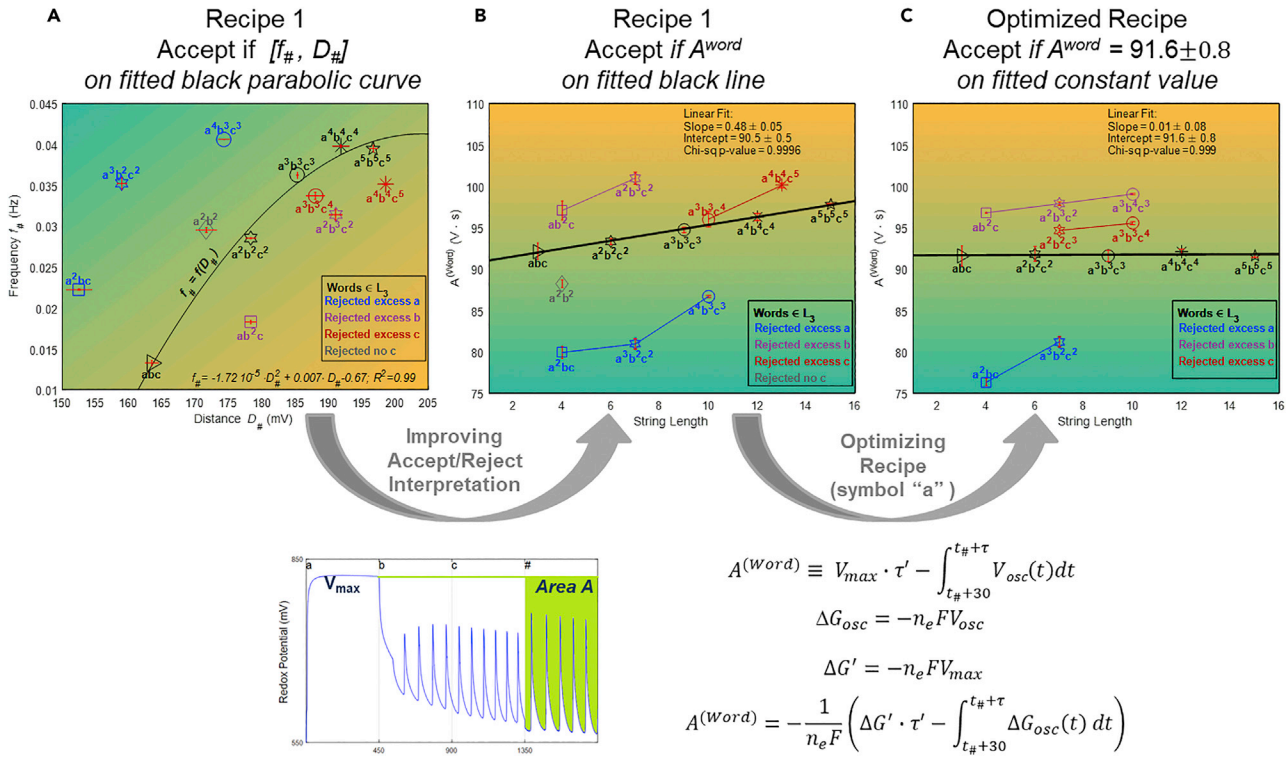


Figure 4. L_3 Language Recognition by the BZ Turing Machine Operating Using a Limited CO_2 Production Recipe (Recipe 1)

Thirteen sequences were each processed three times (cf. Figures S3 and S4) by the L_3-recognizing chemical machine described in the text. In these experiments, eight strings were not in L_3 and five were. (A) shows the completed sequence final frequency $f_{\#}$ vs. final distance $D_{\#}$ for these input strings. (B) and (C) show the evolution of the redox potential during sequence processing for rejected strings $a^3b^2c^2$ and $a^2b^3c^2$ [note how in the latter $f_{\#}$ and $D_{\#}$ are not on the locus of $f(D_{\#})$]. (D) and (E) show the same for accepted words $a^2b^2c^2$ and $a^3b^3c^3$, which are seen to satisfy $f_{\#} = f(D_{\#})$.

words in the machine's language. However, to provide a physico-chemical understanding of these results we introduce an equivalent presentation of the experimental data that makes more direct the interpretation of the Acceptance/Rejection of words in a language by its chemical TM.

As noted earlier, the reaction extent is sequence dependent. We also know that the frequency and amplitude of the redox oscillations in BZ result from chemical competition between the oxidation and reduction subnetworks (pathways) of the reaction (Zhabotinsky, 1964, 2007). We use this feature to design a more direct and visual method to determine sequence Acceptance or Rejection for the TM we are discussing, which also brings out the physical interpretation of a computation by a chemical TM. With this in mind, we introduce a chemical variable related to the flux of chemicals in the reaction as alphabet symbols are processed during a computation by the specific BZ-reaction/TM combination. Then as the letters in a word go through the BZ-TM, we measure, register, and plot the redox potential as a function of time. Finally, for a fully processed word, we obtain the difference in area swept by the highest redox voltage reached during the processing of the letters in the word and the area subtended by the redox potential profile **during only** the processing of the "end-of-word" symbol #, but, in order to allow for the decay of "fast" reaction transients, we excluded the first 30 s of this processing. This area, $\text{Area}^{(\text{Word})}$, cf. Figure 5, equals the product of the #-processing time interval after fast-transient decay, $\tau\prime = (\tau - 30)$ sec, multiplied



$$A^{(Word)} \equiv V_{max} \cdot \tau' - \int_{t_{\#}+30}^{t_{\#}+\tau} V_{osc}(t)dt$$

$$\Delta G_{osc} = -n_e F V_{osc}$$

$$\Delta G' = -n_e F V_{max}$$

$$A^{(Word)} = -\frac{1}{n_e F}\left(\Delta G' \cdot \tau' - \int_{t_{\#}+30}^{t_{\#}+\tau} \Delta G_{osc}(t)\, dt\right)$$

Figure 5. Experimental Results and Interpretation of Our Chemical TM Responses before and after Recipe Optimization to a Zero-Slope Line for Words in the TM-L_3 Combination, and the Relation between Free Energy and Subtended Area $A^{(Word)}$

(A)–(C) display the experimental results for the sequences that are accepted or rejected upon processing the "end-of-word" symbol #. (A) is the bivariate [$f_{\#}$, $D_{\#}$] oscillatory description, (B) is the fully equivalent univariate $A^{(Word)}$ oscillatory description, and (C) shows how after tuning the recipe (by increasing the concentration of aliquot "a") (cf. Transparent Methods and Figures S2 and S5) $A^{(Word)}$ is rendered constant for words in L_3. The bottom panel illustrates the graphical definition of $A^{(Word)}$ as the difference between the area swept by the fully oxidized potential (V_{max}) and the area encompassed by the oscillations after the end-of-word symbol aliquot is added and processed during time τ', together with the physico-chemical description and thermodynamic meaning of $A^{(Word)}$. Note how for the optimized recipe the area is constant and independent of string length only for accepted words. Additionally, rejected strings cluster either above or below this constant value, showing how the Area vs. String Length plot clearly discriminates between the machine's word Acceptance/Rejection regions.

by the free-energy difference between the oxidized and reduced states of the catalyst after the processing of #, i.e., $Area^{(Word)} = \tau\prime \times \Delta G^{(Word)}_{RedOx}$. We note in passing, that the dimensions of this area are those of the action in physics.

To further optimize the acceptance criteria, using computer simulations of BZ we identified for various aliquot recipes which of them had the most impact on the slope of the $Area^{(Word)}$ vs. string length plot (cf. Transparent Methods Section 6 and Figure S2). Then, we determined experimentally the chemical recipe for the symbols in the alphabet, optimized under the condition that all the words in the language accepted by our TM subtend an area that is independent of the language index *n*. In other words, cf. Figures 5 and S5, the free-energy difference $\Delta G^{(Word)}_{RedOx}$ is the same for all words in the language accepted by the chemical TM. Chemically this is because the changes in the extents of oxidation and reduction cancel each other for words in the language accepted by the BZ-recipe/TM/language combination. For words not in L_3, the chemical pathways and the degrees to which they are visited during word processing are such that the oxidation and reduction extents do not compensate and, as a consequence, there is an *n*-dependence in the flux magnitude previously introduced and represented by the value of $Area^{(Word)}$ (cf. Figure 5).

This means that the entropy produced during the processing of the end-of-word symbol for any word in the language accepted by our isothermal chemical Turing machine is such that it compensates in the same amount the internal chemical energy dissipated for every such word and the free energy is, therefore, independent of *n*. This is not true for words not in the machine's language. (Note that the methodology

to tune the acceptance criteria in terms of constant or linear energy dependence on string length can be extended to other Context-Sensitive languages, specifically the family of languages $L = \{a^{n+\alpha}b^{n+\beta}\, c^{n+\gamma}\}$ for $\alpha \geq 0$, $\beta \geq 0$, $\gamma \geq 0$, and $n \geq 1$).

Finally, any chemical automaton, as is the case for its abstract counterpart inclusive hierarchy, should recognize languages at its hierarchy level and lower (Motwani et al., 2010). Indeed, the B-Z TM can recognize L_1 (presence of redOx oscillations = Acceptance and no redox oscillations = Rejection) and the Dyck language L_2 (Pérez-Mercader et al., 2017).

Conclusions

Without using any biochemistry or external auxiliary geometrical constructs, only "one pot" inorganic chemistry, we have implemented experimentally strictly native chemical realizations of automata in the classical automata hierarchy: a Finite Automaton, a one-stack Push-Down Automaton, and a Turing machine (in the Limited Bounded Automaton subclass). Each automaton recognizes a well-defined representative language, L_1, L_2, and L_3, respectively.

As in Automata Theory, the computations consist on processing for Acceptance or Rejection by the chemical automaton of chemically transcribed symbols ordered in a sequence belonging to a language at the appropriate level in the Chomsky hierarchy.

The computations are carried out by the molecules of substances recognizing each other while participating in suitable reactions. They use chemical energy and output the results of the computation through unambiguous physico-chemical signatures of the state of the reaction during and at the end of the computation. The information in a sequence (originally digital information) is transcribed into a sequence of chemical aliquots of specific composition and concentration (analog information), which is examined in parallel, molecule by molecule, by the chemical reaction underlying the automaton. The output of this recognition of analog information is digital and consists of either Acceptance or Rejection of the information fed between the Start- and End-of-word symbol depending on whether said information belongs or does not belong to the language associated with the automaton.

Experimental realization of chemical automata poses challenges since, for example, the monitoring system or specific chemical recipes and mode of operation affect the maximum reliably testable string lengths. These limitations can be overcome by aliquot optimization and by running the BZ reaction in a Continuously Stirred Tank Reactor mode, with the input to be computed still fed sequentially at regular intervals, but with a continuous inflow and outflow of the liquid medium to avoid excessive intermediate accumulation and time-dependent dilution effects.

Since only the oscillatory nature of the reaction is involved (in word Acceptance/Rejection) Chemical TMs recognizing L_3 (or other suitable decidable languages) may be designed using three-reactant non-linear chemical oscillators other than BZ, including DNA or other organic oscillators. The ability to implement different instances of automata translates into practical suggestions, including the use of the automata's chemical output as signals to communicate to the external world the result of a computation using BZ-coupled hydrogels, or feeding the output of one automaton as input to other automata down the line to implement complex information architectures, or to function, for example, as transponders or chemoactivators. These automata use Avogadro's number of processors and therefore we can expect robustness in their operation. The design methodology for the automaton's sequence acceptance criteria based on a free-energy measure opens a path to control the energetics of computation (Bennett, 1982). The free energy spent for every word accepted by this isothermal chemical automaton was rendered constant by an appropriate formulation of the chemical recipe associated with the pair machine-language and, for accepted words, the entropy rate offsets the energy dissipation in the same amount.

The implementation of native chemical computation beyond the level of logic gates (which are themselves Finite Automata [Hopcroft et al., 2007]) is usually assumed to require the complexity of biochemistry. However, our results demonstrate that this widely held belief is not true: chemical computation does not necessitate the presence of life-related chemistry. Chemical computation is a property of matter and not exclusive to life. We also note that chemical computation opens the door for the involvement of baryonic matter in computation, a feature shared with quantum computation.

The fact that chemical automata do not require biochemistry has deep implications for many fields, including the origin of life and advanced biomimetic applications, as it shows that information processing by chemistry is not exclusive to biochemistry, and therefore, albeit in a primitive form, chemical information processing could have existed independently of when DNA/RNA-based life or chemistry appeared on Earth. By increasing the number of stacks and the alphabet length a path emerges where the evolution of chemical complexity in habitable (or proto-habitable) environments brings about a transition in chemistry-based computation, from simple FAs to networks of Turing machines and, eventually, to their combination into more powerful molecular computers. That is, a path for the sequential/hierarchical origin of information handling in nature and the transition from non-biochemical to biochemical and extant life-related computation.

Recently, polymerization-induced self-assembly (PISA) of polymersomes using the radicals produced in BZ has been demonstrated experimentally (Bastakoti and Pérez-Mercader, 2017a, 2017b) showing how, in a suitable environment, BZ can generate its own polymeric enclosure and a corresponding free-energy gradient. This leads to soft microrobots with an entrapped BZ (or another oscillatory chemistry) running as a wet automaton that controls its own encapsulation with a cargo capable of further specific computations, and, more importantly, meaningful functions. These programmable soft robots are now closer to being built ex-novo in a laboratory.

Limitations of the Study

The maximum input word length that we tested experimentally in our Turing machine setup was limited to about 20 letters because of a variety of factors. These factors include the chosen chemical recipes, dilution effects, reactor operation mode, operating conditions, and monitoring method (gas produced during the reaction in the BZ reaction interferes with the precision of the redox monitoring). Further optimization of the chemical recipes, reactor operation mode, and operating conditions, as well as advances in monitoring methods, can provide means to extend this maximum word length.

METHODS

All methods can be found in the accompanying Transparent Methods supplemental file.

SUPPLEMENTAL INFORMATION

Supplemental Information can be found online at https://doi.org/10.1016/j.isci.2019.08.007.

ACKNOWLEDGMENTS

We thank Repsol S. A. for supporting this research. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

AUTHOR CONTRIBUTIONS

Both authors contributed equally to this work.

DECLARATION OF INTERESTS

The authors do not have any conflict of interests. Both authors have a patent on the subject of this paper. The patent is US Patent 9,582,771 B2, February 28, 2017, and is cited as reference 33 of this paper.

Received: April 10, 2019
Revised: July 8, 2019
Accepted: August 2, 2019
Published: September 27, 2019

REFERENCES

Adamatzky, A., and Costello, B.D.L. (2002). Experimental logical gates in a reaction-diffusion medium: the XOR gate and beyond. Phys. Rev. E *66*, 046112.

Adleman, L.M. (1994). Molecular computation of solutions to combinatorial problems. Science *266*, 1021–1024.

Ball, P. (1999). The Self-Made Tapestry: Pattern Formation in Nature (Oxford University Press).

Bar-Hillel, Y., Perles, M., and Shamir, E. (1961). On formal properties of simple phase structured grammars. Z. Phonetik Sprachwiss. Kommunikationsforschung *14*, 143–172.

Bastakoti, B., and Pérez-Mercader, J. (2017a). Facile one pot synthesis of functional giant polymeric vesicles controlled by oscillatory chemistry. Angew. Chem. Int. Ed. *56*, 12086–12091.

Bastakoti, B., and Pérez-Mercader, J. (2017b). Autonomous ex-novo chemical assembly with blebbing and division of functional polymer

vesicles from a "homogeneous mixture". Adv. Mater. *29*, 1704368.

Belousov, B.P. (1959). A periodic reaction and its mechanism. Compilation of Abstracts on Radiation Medicine *147*, 1.

Benenson, Y. (2009). Biocomputers: from test tubes to live cells. Mol. Biosys. *5*, 675–685.

Bennett, C.H. (1982). The thermodynamics of computation—a review. Int. J. Phys. *21*, 905–940.

Brookshear, J.G. (1989). Theory of Computation: Formal Languages, Automata and Complexity (Addison Wesley).

Bryant, B. (2012). Chromatin computation. PLoS One *7*, e35703.

Chomsky, N. (1956). Three models for the description of language. IRE Trans. Inform. Theory *2*, 113–124.

Cohen, D. (1991). Introduction to Computer Theory, Second Edition (John Wiley & Sons, Inc).

Conrad, M. (1972). Information processing in molecular systems. BioSystems *5*, 1–14.

Dueñas-Díez, M., and Pérez-Mercader, J. (2019). Native chemical automata and the thermodynamic interpretation of their experimental accept/reject responses. In The Energetics of Computing in Life and Machines, D.H. Wolpert, C. Kempes, J.A. Grochow, and P.F. Stadler, eds. (SFI Press), pp. 119–139.

Field, R.J., Körös, E., and Noyes, R.M. (1972). Oscillations in chemical systems: thorough analysis of temporal oscillation in the bromate-cerium-malonic acid system. J. Am. Chem. Soc. *94*, 8649–8664.

Floyd, R.W., and Beigel, R. (1994). The Language of Machines: An Introduction to Computability and Formal Languages (Computer Science Press, Inc.).

Gentili, P.L., Horvath, V., Vanag, V.K., and Epstein, I.R. (2012). Belousov-zhabotinsky "chemical neuron" as a binary and fuzzy logic processor. IJUC *8*, 177–192.

Gorecki, J., Yoshikawa, K., and Igarashi, Y. (2003). On chemical reactors that can count. J. Phys. Chem. A *107*, 1664–1669.

Harrison, M.A. (1978). Introduction to Formal Language Theory, First Edition (Addison-Wesley Longman Publishing Co., Inc.).

Hjelmfelt, A., Weinberger, E.D., and Ross, J. (1991). Chemical implementation of neural networks and Turing machines. Proc. Natl. Acad. Sci. U S A *88*, 10983–10987.

Hopcroft, J.E., Motwani, R., and Ullman, J.D. (2007). Introduction to Automata Theory, Languages, and Computation, Third Edition (Pearson Education Inc.).

Katz, E. (2012). Molecular and Supramolecular Information Processing. From Molecular Switches to Logic Systems (Wiley-VCH Verlag GmbH & Co. KGaA).

Linz, P. (2012). An Introduction to Formal Languages and Automata, Fifth Edition (Jones & Bartlett Learning).

Lloyd, S. (1992). Any nonlinear gate, with linear gates, suffices for computation. Phys. Lett. A. *167*, 255–260.

Magnasco, M.O. (1997). Chemical kinetics is Turing universal. Phys. Rev. Lett. *78*, 1190.

Minsky, M.L. (1961). Recursive unsolvability of Posts problem of tag and other topics in theory of turing machines. Ann. Math. *74*, 437–455.

Minsky, M.L. (1967). Computation: Finite and Infinite Machines (Prentice Hall).

Moore, C., and Mertens, S. (2011). The Nature of Computation (OUP Oxford).

Motwani, R., Raghavan, P., Atallah, M.J., and Blanton, M. (2010). Algorithms and Theory of Computation Handbook (Chapman & Hall/CRC).

Oettinger, A.G. (1961). Automatic syntactic analysis and the pushdown store. Proc. Sympos. Appl. Math. *12*, 104–129.

Okamoto, M., Sakai, T., and Hayashi, K. (1987). Switching mechanism of a cyclic enzyme system: role as a 'chemical diode'. Biosystems *21*, 1–11.

Pérez-Mercader, J., Dueñas-Diez, M., and Case, D. (2017). Chemically-Operated Turing Machine. US Patent 9,582,771 B2, February 28, 2017.

Petrucci, R.H., Herring, G., Madura, J.D., and Bissonette, C. (2011). General Chemistry: Principles and Modern Applications, Tenth Edition (Pearson Prentice Hall).

Prohaska, S.J., Stadler, P.F., and Krakauer, D.C. (2010). Innovation in gene regulation: the case of chromatin computation. J. Theor. Biol. *265*, 27–44.

Rich, E. (2008). Automata, Computability, and Complexity. Theory and Applications (Pearson/Prentice-Hall).

Searls, D.B. (2012). A primer in macromolecular linguistics. Biopolymers *99*, 203–217.

Soloveichik, D., Seelig, G., and Winfree, E. (2010). DNA as a universal substrate for chemical kinetics. Proc. Natl. Acad. Sci. U S A *107*, 5393–5398.

Sudkamp, T.A. (2006). Languages and Machines: An Introduction to the Theory of Computer Science, Third Edition (Pearson Education, Inc.).

Tóth, A., and Showalter, K. (1995). Logic gates in excitable media. J. Chem. Phys. *103*, 2058–2066.

Turing, A.M. (1936). On computable numbers with an application to the entscheidungs-problem. Proc. Lond. Math. Soc. *2*, 230–265.

Tyson, J.J. (1985). A quantitative account of oscillations, bistability, and travelling waves in the Belousov-Zhabotinskii reaction. In Oscillations and Travelling Waves in Chemical Systems, R.J. Field and R. Burger, eds. (John Wiley), pp. 92–144.

Wang, A.L., Gold, J.M., Tompkins, N., Heymann, M., Harrington, K.I., and Fraden, S. (2016). Configurable NOR gate arrays from Belousov-Zhabotinsky micro-droplets. Eur. Phys. J. Spec. Top. *225*, 211–227.

Weisstein, E.W. (2009). CRC Encyclopedia of Mathematics, Third Edition (CRC Press).

Winfree, A.T. (1984). The prehistory of the Belousov-Zhabotinsky oscillator. J. Chem. Educ. *61*, 661–663.

Zhabotinsky, A.M. (1964). Periodical oxidation of malonic acid in solution (a study of the Belousov reaction kinetics). Biofizika *9*, 306–311.

Zhabotinsky, A.M. (2007). Belousov-Zhabotinsky reaction. Scholarpedia *2*, 1435.

ISCI, Volume *19*

Supplemental Information

How Chemistry Computes: Language Recognition by Non-Biochemical Chemical Automata. From Finite Automata to Turing Machines

Marta Dueñas-Díez and Juan Pérez-Mercader

Supplemental Information (SI)

SUPPLEMENTAL FIGURES

TRANSPARENT METHODS

SUPPLEMENTAL FIGURES

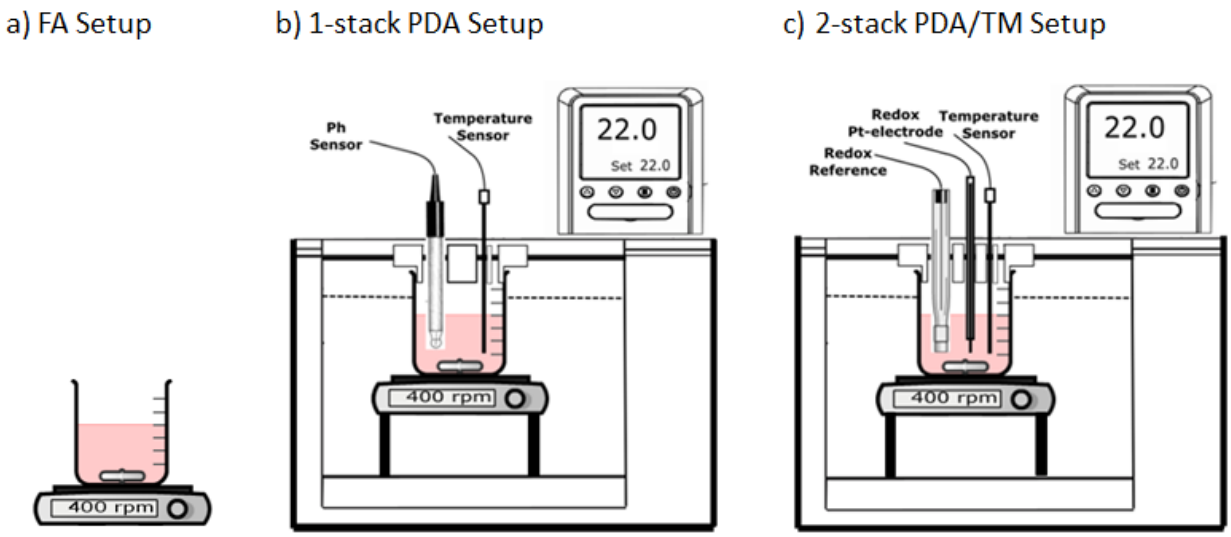


Figure S1: Title: Experimental setup configurations for the three implemented chemical automata, Related to Figures 2, 4 and 5, and to Supplemental Video. Caption: Panel a) shows the simple setup for the chemical FA. Panels b) and c) show the setups for the 1 stack PDA and the TM, respectively. In these two, the beaker was immersed in a thermal bath to keep the temperature constant at 22 Celsius, and a submersible magnetic stirrer was used to keep the solution in the beaker well mixed. Monitoring of key observables was carried out, pH monitoring and temperature in the case of the 1-stack PDA and redox potential and temperature in the case of the chemical TM.

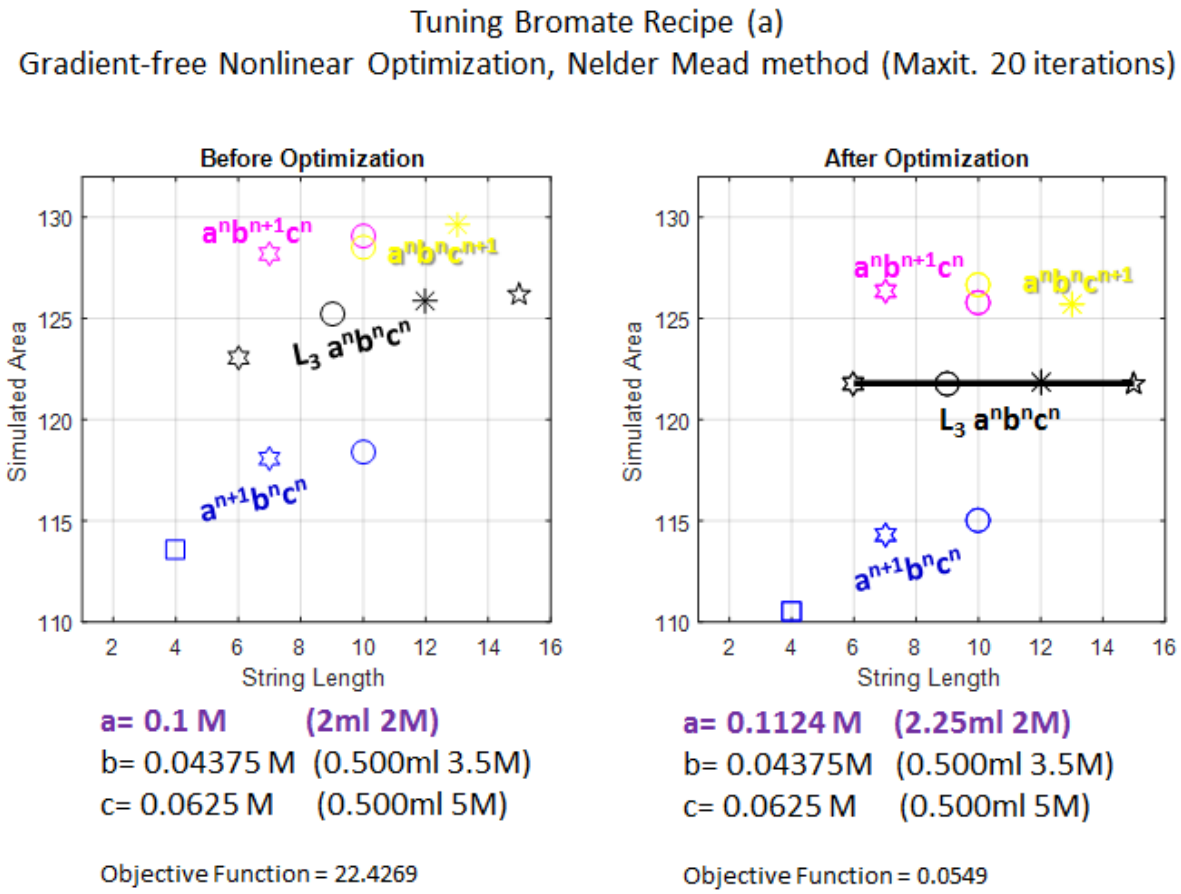


Figure S2: Title: Model-based mathematical optimization of the L_3-TM to render the area constant for words in the language, Related to Figure 5. Caption: Simulations of Area vs. string length before and after mathematical optimization for sequences tested experimentally. For both recipes, the rejected strings are clustered relative to the canonical language L_3 in an equivalent relative position (above or below L_3). The mathematical optimization shows that by just increasing the concentration of aliquot “a” it is possible to make the area independent of string length for words in L_3.

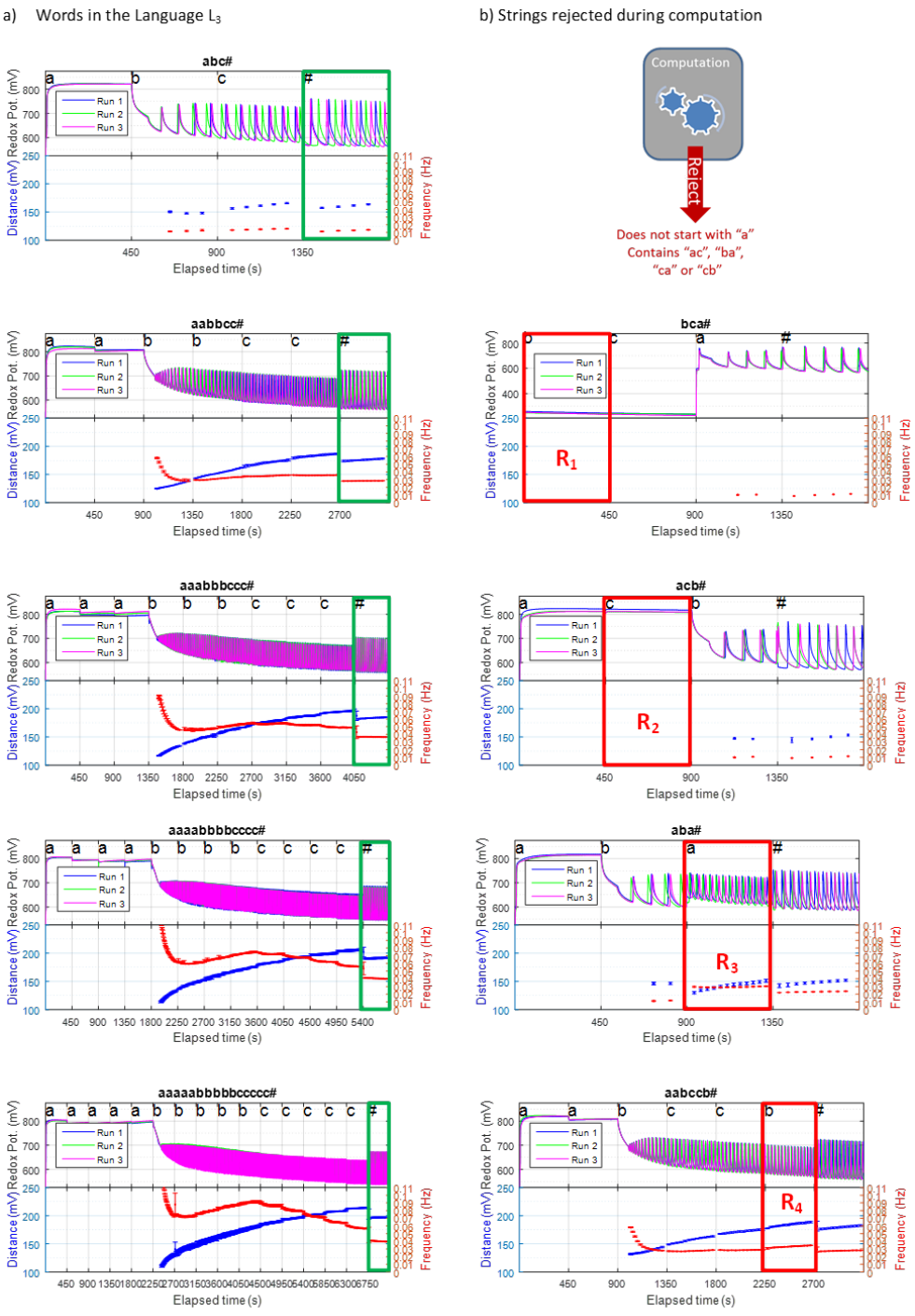


Figure S3: Title: Experimental sequences using aliquot recipe 1 in the BZ-TM recognizing L_3 (accepted sequences and rejected sequences due to incorrect letter order), Related to Figure 4 and to Supplemental Video. Caption: The plots on the a) column show accepted words in language L_3 for n=1, 2, 3, 4 and 5, from top to bottom, respectively. The top two plots in the b) column show rejected strings even before oscillation takes place, *bca* is rejected since it does not start with *a,* while *acb* is rejected because it contains substring *ac*. The bottom two plots in the b) column show rejected strings once in the oscillatory regime due to forbidden substrings, hence *aba* is rejected because it contains substring *ba* and *aabccb* because it contains substring *cb*.

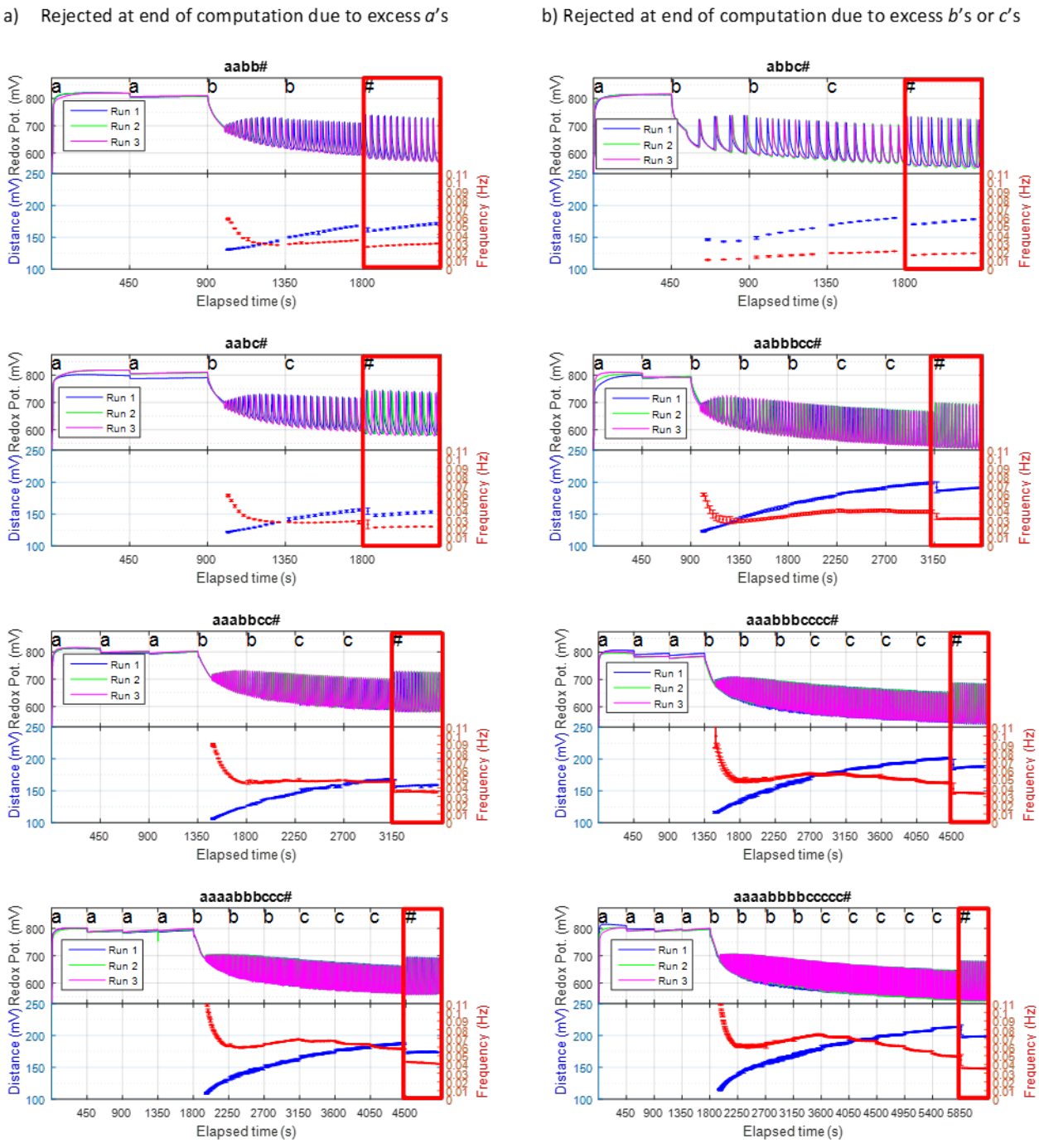


Figure S4: Title: Experimental sequences using aliquot recipe 1 in the BZ-TM recognizing L_3 (rejected sequences due to incorrect letter count at the end of computation), Related to Figure 4. Caption: The plots in the a) column show 4 rejected words due to excess *a*'s, showing a higher final frequency *f* and lower final distance *D* than the nearest word in L_3, e.g. compare *aabc* with *abc* (in Figure S3). The plots in the b) column show 2 rejected words due to excess *b*'s, that are displaced towards higher *f* and *D* with respect to accepted words. The bottom two plots in b) column show 2 rejected words due to excess *c*'s, which are displaced towards lower final *f* and higher *D*.

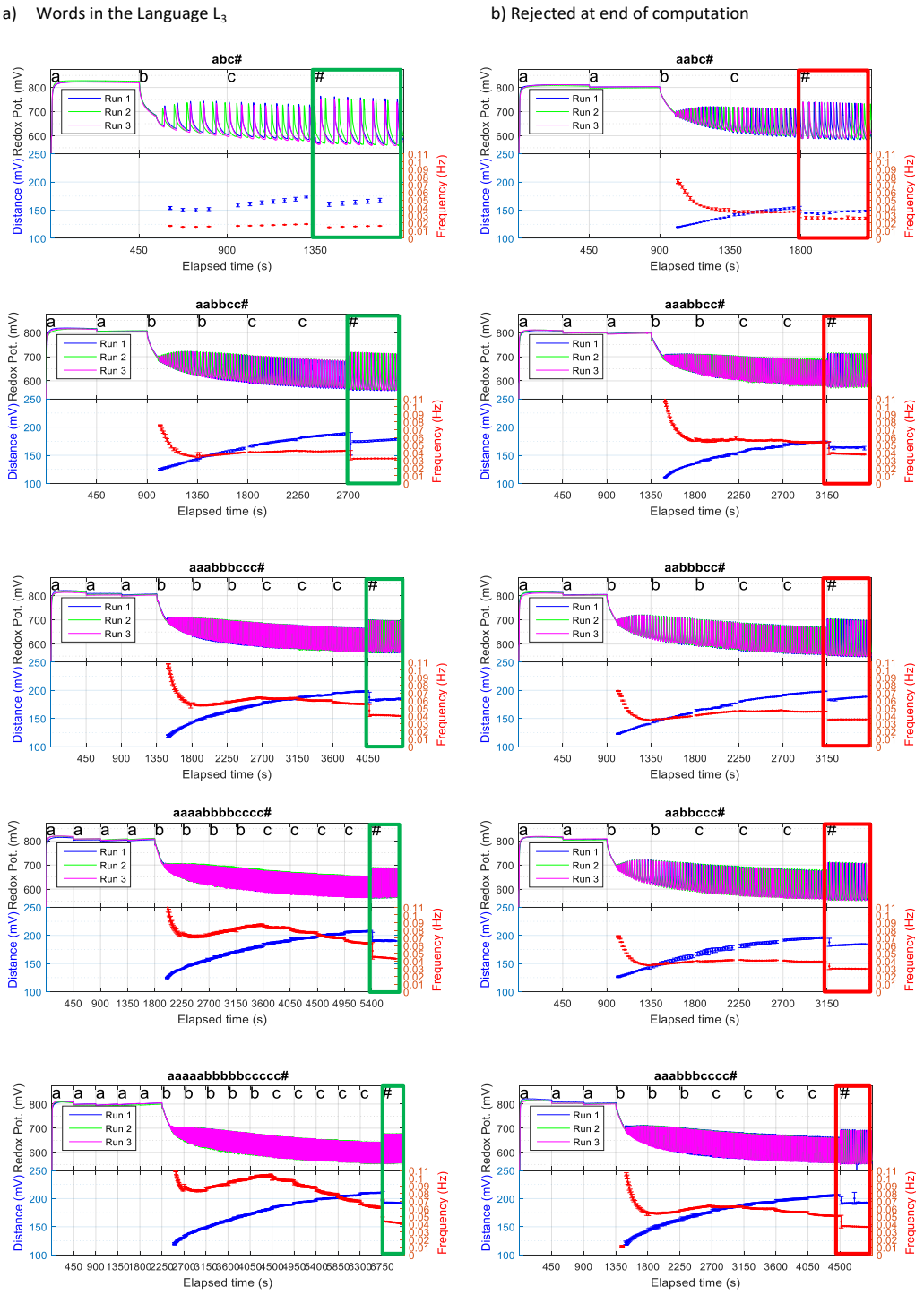


Figure S5: Title: Experimental sequences using aliquot recipe 2 in the BZ-TM recognizing L_3 (accepted and rejected sequences), Related to Figure 5. Caption: The plots on the a) column show the accepted words in language L_3 for n=1, 2, 3, 4 and 5, from top to bottom, respectively. The plots on the b) column show rejected inputs at the end of computation: 2 due to excess *a*'s, 1 due to excess *b*'s and 2 due to excess *c*'s. Compared to the results with recipe 1 (Figures S3 and S4), the dynamic response features are qualitative

equivalent, and quantitatively most are displaced to higher frequencies *f* and lower distance *D* compared to the same input with recipe 1.

TRANSPARENT METHODS

Section 1: Materials and methods for the experimental implementation of a Chemical FA based on the precipitation of silver iodate

Reaction: A chemical realization of this FA is provided by the precipitation reaction:

$$KIO_3 + AgNO_3 \rightarrow AgIO_3\,(s) + KNO_3\,.$$

If during the computation a white precipitate of silver iodate is observed, the input string has been Accepted, whereas if at the end of the string the solution is free from precipitate the string has been Rejected.

Engineering Considerations for recipe and operation design: The main criterion used to choose the recipes for the aliquots was to ensure that the solubility product constant of silver iodate was exceeded once at least one aliquot of KIO_3 and one aliquot of $AgNO_3$ had been added to the reactor. In addition, they were chosen such that the amount of precipitate would be large enough to be visible to the naked eye. Since this precipitation reaction has fast kinetics, any time interval exceeding 5 seconds would suffice for the computation. In our implementation a time interval of 30 seconds was used for execution convenience.

Materials: Commercially available Potassium Iodate KIO_3 (Sigma Aldrich) and Silver Nitrate $AgNO_3$ (Sigma Aldrich) were used without further purification. Reverse-osmosis deionized water (12 Megaohm) was used to prepare the following stock solutions: 0.35 M KIO_3 and 0.5 M $AgNO_3$.

Initial Conditions: The reactor/beaker at the beginning of the computation contained 20 ml of reverse-osmosis deionized water.

Alphabet Recipes: Each "***a***" in the input sequence was implemented by pipetting into the reactor a 1.5±0.03 mL aliquot of a 0.35 M KIO_3 stock solution (Eppendorf Research Plus pipette 1-5 mL). Each "***b***" in the input sequence was implemented by pipetting into the reactor a 1.0±0.02 mL aliquot of 0.5 M $AgNO_3$stock solution (Eppendorf Research Plus pipette 1-5 mL). The input string was pipetted to the reactor sequentially, symbol by symbol, at fixed time intervals of 30±2 s.

Reactor: The experiments were carried out in a semibatch reactor at ambient temperature conditions. A 50 mL volume and 40 mm diameter Pyrex® glass beaker with the 20 mL initial solution was stirred at 400 rpm with a Teflon-coated magnetic stirbar (VWR® Spinbar® Polygon 6.4 x 25 mm) and a digital magnetic stirrer (IKA® Color Squid). During the experiment, the color of the solution was monitored visually for the appearance of a precipitate. A schematic representation of this simple experimental setup is shown in panel a) of Figure S1.

Experiment execution: The initial solution was stirred for ten minutes to achieve thermal equilibrium with the surroundings. At the instant the chronometer was started, the first symbol of the input string was pipetted into the reactor. Then, each of the subsequent symbols in the input string was pipetted to the reactor sequentially, symbol by symbol, at regular time intervals of 30 s. Once the input string had been completed, and the last time interval of 30 s had finished, the state of the reactor contents showed the machine's output. If the solution in the beaker was still transparent the chemical machine had rejected the input string, while if the solution had a white color (presence of silver iodate precipitate), then the chemical FA had reached the Accept state, and the input string was accepted as a word in the language. Alternatively, the heat of reaction could have been monitored by calorimetry, and if while the chemical computation any measurable heat of reaction had been detected, then this would indicate reaching the accept state. Otherwise, if no measurable heat of reaction was detected, the machine had rejected the input string.

Section 2: Materials and methods for the experimental implementation of a chemical 1-stack PDA based on pH chemistry

Reaction: For our chemical realization of the parenthesis checker we use the following pH reaction network:

$$C_3H_4O_4 \rightarrow C_3H_3O_4^- + H^+ \quad (R_1)$$
$$C_3H_3O_4^- \rightarrow C_3H_2O_4^{2-} + H^+ \quad (R_2)$$
$$H^+ + OH^- \rightarrow H_20 \quad (R_3)$$
$$HInd + OH^- \rightarrow Ind^- + H_2 0 \quad (R_4)$$
$$H^+ + Ind^- \rightarrow HInd \quad (R_5)$$

Where Methyl Red (2-(N,N-dimethyl-4-aminophenyl) azobenzenecarboxylic acid) is used as pH Indicator, whose chemical formula is $C_{15}H_{15}N_3O_2$

Engineering Considerations for recipe and operation design: We chose a polyprotic acid, malonic acid, because it makes computation more robust. If we had used a strong acid and a strong base, the recipes to achieve compensation of parentheses should have been adjusted to the equivalence point instead, and it is well known from pH titration that even small errors on the aliquots would lead the solution to either a high or low pH, instead of the equivalence point. Meanwhile, for a polyprotic acid the compensation of parentheses pairs can be designed to match with a midpoint instead of an equivalent point, which is not sensitive to small errors in the aliquots. In our implementation, the aliquots of acid and base are chosen such that when one aliquot of ***(*** and one aliquot of ***)*** are added to the solution the pH corresponds to the midpoint after the first equivalence point, hence the point at which $HOOCCH_2COO^-$ is in chemical equilibrium with $OOCCH_2COO^{2-}$. Methyl Red was chosen as the end-of-expression symbol since it is a pH indicator that changes color in between the first and the second equivalence points, therefore producing different solution colors for input strings that are accepted than for those that are rejected, and with even distinct colors for each of the two types of reject, yellow for excess open and magenta for excess closed parentheses. Acid-base neutralization reactions have fast dynamics, so a time interval in the order of tenths of seconds would suffice, but we chose 3 min for convenience and precision in the experimental execution.

Materials: Commercially available Sodium Hydroxide $NaOH$ (Amresco) and Malonic Acid $CH_2(COOH)_2$ (Alfa Aesar) were used without further purification to prepare stock solutions. Commercially available Methyl-Red (2-(N,N-dimethyl-4-aminophenyl) azobenzenecarboxylic acid (Sigma Aldrich, Solution pH) was used as the pH indicator. Reverse-osmosis deionized water (12 Megaohm) was used to prepare the following stock solutions: 4.9 M $NaOH$, 3.5 M $CH_2(COOH)_2$.

Initial Conditions: The reactor/beaker at the beginning of the computation contained 40 ml of reverse-osmosis deionized water, and an aliquot of Methyl Red pH indicator.

Alphabet Recipes: For each open parenthesis "***(***" in the input string, a 0.535±0.004 mL aliquot of 4.9 M NaOH stock solution was pipetted (Eppendorf Research Plus pipette 0.1-1 mL) into the reactor. For each closed parenthesis "***)***" a 0.500±0.004 mL aliquot of 3.5 M $CH_2(COOH)_2$ stock solution was pipetted (Eppendorf Research Plus pipette 0.1-1 mL) into the reactor. A symbol "**#**' is needed in the PDA and TM to indicate the beginning and end of the input string, here it was implemented with a 0.300±0.002 mL aliquot of the commercial Methyl Red (Eppendorf Research Plus pipette 0.1-1 mL). The input string was pipetted to the reactor sequentially, symbol by symbol, at regular time intervals of 180±2 s.

Reactor: The experiments were carried out in a semibatch reactor under controlled temperature conditions. A 100 mL volume and 50 mm diameter Pyrex® glass beaker with the 40 mL initial solution was submerged in a 7L refrigerated circulating bath (VWR MX7LR) at a constant setpoint of 22.0 °C. The reaction mixture was stirred at 400 rpm with a Teflon-coated magnetic stirbar (VWR® Spinbar® Polygon 6.4 x 35 mm) and a submersible magnetic stirrer (2Mag Mixdrive 1 eco and 2Mag Mixcontrol eco). The change in pH of the

reaction mixture was monitored with a commercial pH meter (SperScientific pH probe 850059P) connected to a benchtop meter (SperScientific). The temperature in the solution was followed by a temperature sensor included in the pH meter and was maintained during computation by the circulating thermal bath at 22.0±0.3 °C. Voltage and pH data were recorded with Labview Signal Express at a frequency of 5 data points per second. A schematic representation of the experimental setup is shown in panel b) of Figure S1.

Experiment execution: The experiment started when the reactor with the initial solution of water was immersed in the thermal bath. The first ten minutes were used to achieve thermal equilibrium between the beaker and the bath temperature. Then, an aliquot of methyl red was pipetted into the reactor, and 180s were allowed for stabilization. The sequence checking procedure starts by pipetting the first symbol in the input sequence, and this is when the computation time starts. All subsequent symbols were pipetted sequentially, symbol by symbol, at exactly 180s intervals. If at any moment during computation the pH reached acidic conditions below the midpoint pH, the solution would turn magenta indicating that the chemical machine rejected the input string. Otherwise, the whole string would be processed, the end-of-expression pipetted and after the last time interval of 180 s elapses, the final status of pH and/or color of the solution provided the machine's answer. If the pH was basic (pH over midpoint pH), i.e. the solution color was yellow, the machine had rejected the input string. Alternatively, if the pH was at the midpoint value, i.e. the solution color was orange, the chemical machine accepted the input string as a word in the language. For each input string, the experiment was run three times, and the pH trends recorded and stored.

Section 3: Materials and methods for the experimental implementation of a Chemical Turing Machine based on Belousov-Zhabotinsky Reaction

Materials. Commercially available Sodium Bromate $NaBrO_3$ (Alfa Aesar), Malonic Acid $CH_2(COOH)_2$ (Alfa Aesar), Tris(2,2'-bipyridyl) Dichloro Ruthenium(II) Hexahydrate $Ru(bpy)_3Cl_2(6H_2O)$ (Sigma Aldrich), Sodium Hydroxide $NaOH$ (Amresco) and Sulfuric Acid solution H_2SO_4 (10N/5M, Fisher Chemical) were used without further purification. Reverse osmosis deionized water (12 Megaohm) was used to prepare the following stock solutions: 2 M $NaBrO_3$, 3.5 M $CH_2(COOH)_2$, 5 M $NaOH$ and 0.0125 M $Ru(bpy)_3Cl_2(6H_2O)$.

Initial conditions. The initial solution was prepared by mixing 33.20 mL of deionized water and 6 mL of 5 M Sulfuric Acid solution, and an aliquot of Ruthenium catalyst (**#**).

Recipe 1 (Maximal word length without disturbance of the redox potential measurement). The canonical L_3 TM language is based on an alphabet of three letters {***a***, ***b***, ***c*** }={$NaBrO_3$, $CH_2(COOH)_2$, $NaOH$}. For each "***a***" in the input sequence a 2.0 ±0.03 mL aliquot of 2.0 M $NaBrO_3$ stock solution was pipetted into the reactor (Eppendorf Research Plus pipette 0.5-5 mL), thus incrementing the bromate concentration in the reactor by 0.10M. For each "***b***" in the input sequence, a 0.500±0.004 mL aliquot of 3.5 M stock malonic acid solution was pipetted into the reactor (Eppendorf Research Plus pipette 0.1-1 mL), hence incrementing the malonic acid concentration in the reactor by 0.04375 M. For each "***c***" in the sequence, 0.500±0.004 mL aliquot of 5 M stock sodium hydroxide was pipetted into the reactor (Eppendorf Research Plus pipette 0.1-1 mL). The beginning and end-of-expression symbol "**#**" was implemented as a 0.800±0.006 mL aliquot of 0.0125 M stock ruthenium complex solution (Eppendorf Research Plus pipette 0.1-1 mL) which increases the catalyst concentration in the reactor by 0.00025 M. The input string was pipetted to the reactor sequentially, symbol by symbol, at regular time intervals of 450±2 s.

Reactor: The experiments for the BZ-TM to recognize language L_3 were carried out in a semibatch reactor under controlled temperature conditions. A 100 mL volume and 50 mm diameter Pyrex® glass beaker with the initial solution was submerged in a 7L refrigerated circulating bath (VWR MX7LR) at a constant setpoint of 22.0 °C. The reaction mixture was stirred at 400 rpm with a Teflon-coated magnetic stirbar (VWR® Spinbar® Polygon 6.4 x 35 mm) and a submersible magnetic stirrer (2Mag Mixdrive 1 eco and 2Mag Mixcontrol eco). The change in the oxidation-reduction (redox) potential of the reaction mixture was monitored with an electrode system composed of a Pt-working electrode and a mercury sulfate reference electrode (Koslow 5100 A) connected to a Vernier Electrode Amplifier (EA-BTA) and to a Vernier

SensorDAQ Data Acquisition Box. The temperature in the solution was monitored with an RTD sensor probe (Omega PR-13-2-100-1 and signal conditioner RTD SPRTX-S1) and was maintained by the circulating bath at 22.0±0.3 °C during the experiment. Redox and temperature data were recorded with Labview® at a frequency of 5 data points per second. The refrigerated circulating bath opening was covered with aluminum foil to avoid light interferences since the used catalyst is photosensitive. A schematic representation of the experimental setup is shown in panel c) of Figure S1.

Experimental Realization: The experiment started when the beaker with the initial solution was immersed in the thermal bath. The first ten minutes were used to achieve thermal equilibrium between the beaker and the bath temperatures. Then, an aliquot of catalyst (0.800±0.006 mL of 0.0125 M stock ruthenium complex solution) was pipetted, indicating the beginning of the input sequence. The sequence checking procedure starts 450s later by pipetting the first symbol in the input sequence, and this is the instant computation time starts. All subsequent symbols were pipetted sequentially, symbol by symbol, at 450s intervals. The precision of the additions of the aliquots was within ±2 s. If at any moment during computation the behavior of the redox potential matches the trend of any of the reject states, then the chemical machine would be rejecting the input string. Otherwise, the whole string would be processed, the end-of-expression pipetted and during the last time interval of 450 s, the final features of the redox oscillation would be evaluated, and the input would be accepted or rejected according to the acceptance criteria.

Monitoring and Data Analysis: In the BZ-based L_3={$a^n b^n c^n$, n>0} language, the relaxation oscillations in the redox potential were monitored. For each string to be computed, the experiment was repeated three times. The recorded data were analyzed, visualized and plotted in Matlab® (R2015b v. 8.6.0.267246). Matlab® Signal Processing Toolbox was used to analyze the oscillation features of the data for each of the three individual repetitions of the string: the peaks and the troughs of the individual oscillations were detected for the recorded time series. The period of the oscillations was defined as the elapsed time difference between two consecutive oscillation peaks, the frequency as the inverse of the period, and the trough-to-peak amplitude as the difference between the redox potential of a peak and the redox potential of its subsequent trough.

For L_3, the distance *D* between the maximum redox value (i.e. when all catalyst is in the oxidized form) and the redox value of the center of a given oscillation is a more adequate feature than amplitude, i.e. we define the distance *D* as:

$$D = V_{max} - \left(V_T + \frac{V_P - V_T}{2}\right)$$

where V_{max} corresponds to the redox potential of all the catalyst in oxidized form, V_T is the redox value of the trough of the oscillation, and $V_P - V_T$ is the trough-to-peak amplitude of the redox oscillation, hence $V_T + \frac{V_P - V_T}{2}$ is the redox value of the center of the oscillations.

Error bars were then estimated as symmetric error bars in the form of: mean ± standard deviation, where the standard deviation was normalized by the number of observations (i.e. 3). Hence, the mean final frequency $f_\#$ and the standard deviation of the final frequency $std(f_\#)$, the mean final distance $D_\#$, the standard deviation of the final distance $std(D_\#,)$ were estimated respectively as:

$$f_\# = \frac{\sum_{i=1}^{3} f_i}{3}$$

$$D_\# = \frac{\sum_{i=1}^{3} V_{max,i}}{3} - \frac{\sum_{i=1}^{3} V_{center,i}}{3}$$

$$std(f_\#) = \sqrt{\frac{\sum_{i=1}^{3}(f_i - f_\#)^2}{3}}$$

$$std(D_\#) = \sqrt{\frac{\sum_{i=1}^{3}(D_i - D_\#)^2}{3}}$$

And the error bars for the amplitude e_A , frequency e_f and for the distance e_D, are given respectively by:

$$e_f = f_\# \pm std(f_\#)$$
$$e_D = D_\# \pm std(D_\#)$$

The area $A^{(Word)}$ was integrated approximately via the trapezoidal method with spacing 0.2 s (same as sampling interval in the data) by means of the trapz command in Matlab®. Note that for this integration we have used the last 7 min of the 7.5 min interval after the end-of-expression symbol was added. Again, the error bars for the area are given as the mean value plus/minus one standard deviation: $e_A = A^{(Word)} \pm std\left(A^{(Word)}\right)$.

Section 4: Extended FKN Mechanism of Belousov-Zhabotinsky

In order to design and optimize the recipes of our implementation of the chemical TM, a representative mathematical model of the kinetic network is needed. The mechanism of the BZ reaction is complex, and there is a variety of kinetic models ranging from small models with 5 elementary steps to very detailed mechanisms. In our initial work on chemical automata, the first validations were based on simulation studies using one of the simplest and yet one of the most widely used mechanistic models of the BZ reaction, called Oregonator. This model proved the concept and ideas. However, in order to capture more realistically the time constants, decay trends and the comparison to automata theory, a more comprehensive model was needed. We tested the Field-Körös-Noyes[38] (FKN) and the Gao-Försterling (GF) models since these are recognized to be better approximations of the observed behavior. Both gave satisfactory results. The FKN model was finally chosen since it is simpler and yet representative enough. The FKN reaction network is comprised of the equations below

$\mathrm{HOBr} + \mathrm{Br^-} + \mathrm{H^+} \leftrightarrow \mathrm{Br_2} + \mathrm{H_2O}$	r_{1a}, r_{1b}
$\mathrm{HBrO_2} + \mathrm{Br^-} + \mathrm{H^+} \rightarrow 2\mathrm{HOBr}$	r_2
$\mathrm{BrO_3^-} + \mathrm{Br^-} + 2\mathrm{H^+} \rightarrow \mathrm{HOBr} + \mathrm{HBrO_2}$	r_3
$2\mathrm{HBrO_2} \rightarrow \mathrm{HOBr} + \mathrm{BrO_3^-} + 2\mathrm{H^+}$	r_4
$\mathrm{HBrO_2} + \mathrm{BrO_3^-} + \mathrm{H^+} \leftrightarrow 2\mathrm{BrO_2} + \mathrm{H_2O}$	r_{5a}, r_{5b}
$[\mathrm{Ru(bpy)_3^{2+}}] + \mathrm{BrO_2} + \mathrm{H^+} \rightarrow [\mathrm{Ru(bpy)_3^{3+}}] + \mathrm{HBrO_2}$	r_6
$\mathrm{CH_2(COOH)_2} + \mathrm{HOBr} \rightarrow \mathrm{CHBr(COOH)_2} + \mathrm{H_2O}$	r_7
$\mathrm{CH_2(COOH)_2} + \mathrm{Br_2} \rightarrow \mathrm{CHBr(COOH)_2} + \mathrm{Br^-} + \mathrm{H^+}$	r_8
$\mathrm{CH_2(COOH)_2} + 6[\mathrm{Ru(bpy)_3^{3+}}] + 2\mathrm{H_2O} \rightarrow \mathrm{HCOOH} + 6[\mathrm{Ru(bpy)_3^{2+}}] + 2\mathrm{CO_2} + 6\mathrm{H^+}$	r_9
$\mathrm{CHBr(COOH)_2} + 4[\mathrm{Ru(bpy)_3^{3+}}] + 2\mathrm{H_2O} \rightarrow \mathrm{HCOOH} + 4[\mathrm{Ru(bpy)_3^{2+}}] + 2\mathrm{CO_2} + \mathrm{Br^-} + 5\mathrm{H^+}$	r_{10}

This model simulates quite properly the experiments, except for expressions with an excess of sodium bromates due to its simplified treatment of the malonic bromination kinetics. For most of the experimental and simulated conditions, the overall malonic bromination kinetics are limited by r_8, more precisely by the bromine concentration, and the FKN gives realistic results. However, when the bromine concentration becomes large enough (as when there is excess of bromates and # is added) then the limiting kinetic step is no longer reaction r_8 and the overall bromination kinetics become zeroth order with respect to bromine.

We modified the FKN based on knowledge gathered from the GF model simulations, extending the model with the following reaction:

$$\mathrm{CHBr(COOH)_2 + Br_2 \rightarrow CBr_2(COOH)_2 + Br^- + H^+} \quad r_{11}$$

After this extension, the model was able to simulate more realistically all the experiments.
Using the following notation

$$\mathrm{x_1 = [HBrO_2]} \qquad \mathrm{x_2 = [Br^-]} \qquad \mathrm{x_3 = [Ru(bpy)_3^{3+}]}$$
$$\mathrm{x_4 = [BrO_3^-]} \qquad \mathrm{x_5 = [CH_2(COOH)_2]} \qquad \mathrm{x_6 = [HOBr]}$$
$$\mathrm{x_7 = [Br_2]} \qquad \mathrm{x_8 = [BrO_2]} \qquad \mathrm{x_9 = [CHBr(COOH)_2]}$$
$$\mathrm{x_{10} = [Ru(bpy)_3^{2+}]} \qquad \mathrm{x_{11} = [CO_2]} \qquad \mathrm{x_{12} = [CBr_2(COOH)_2]}$$
$$\mathrm{H = [H^+]}$$

Most kinetic rates follow the Mass Action law and the kinetic constants, measured at 20 °C, are taken from (Vanag & Epstein (2008). The kinetic constant for reaction r_{11} has been estimated from the GF-model.

$$v_{1a} = k_{1a}\mathrm{x_2x_6H} \quad k_{1a} = 5e^9\ \mathrm{M^{-2}}s^{-1}$$
$$v_{1b} = k_{1b}\mathrm{x_7} \quad k_{1b} = 10\ s^{-1}$$
$$v_2 = k_2\mathrm{x_1x_2H} \quad k_2 = 2e^6\ \mathrm{M^{-2}}s^{-1}$$
$$v_3 = k_3\mathrm{x_2x_4H^2} \quad k_3 = 2\mathrm{M^{-3}}s^{-1}$$
$$v_4 = k_{4a}\mathrm{x_1}^2 \quad k_4 = 3e^3\ \mathrm{M^{-1}}s^{-1}$$
$$v_{5a} = k_{5a}\,\mathrm{x_1x_4H} \quad k_{5a} = 42\ \mathrm{M^{-2}}s^{-1}$$
$$v_{5b} = k_{5b}\,\mathrm{x_8}^2 \quad k_{5b} = 2e^8\ \mathrm{M^{-1}}s^{-1}$$
$$v_6 = k_6\mathrm{x_8x_{10}H} \quad k_6 = 3e^6\ \mathrm{M^{-2}}s^{-1}$$
$$v_7 = k_7\mathrm{x_5x_6H} \quad k_7 = 9.3\ \mathrm{M^{-2}}s^{-1}$$
$$v_8 = k_8\mathrm{x_5x_7} \quad k_8 = 29\ \mathrm{M^{-1}}s^{-1}$$
$$v_9 = k_9\,\mathrm{x_3x_5} \quad k_9 = 5e^{-2}\ \mathrm{M^{-1}}s^{-1}$$
$$v_{10} = k_{10}\,\mathrm{x_3x_9} \quad k_{10} = 7\ \mathrm{M^{-1}}s^{-1}$$
$$v_{11} = k_8\mathrm{x_9x_7} \quad k_{11} = 29\ \mathrm{M^{-1}}s^{-1}$$

For the simulation of the languages L_3, where *c* is an aliquot of NaOH, the model needs to be extended by including the acid-base neutralization reaction:

$$\mathrm{OH^- + H^+ \rightarrow H_2O} \quad r_{12}$$

Since the proton concentration is in large excess with respect to the amount of NaOH added with the aliquot, we assume in the model that all $\mathrm{OH^-}$ is instantaneously and completely consumed, decreasing the proton concentration $\mathrm{H^+}$ and thus affecting all the rate constants that depend on the proton concentration $\mathrm{H^+}$. Note that the bromine chemistry and the oxidation reactions are the ones most dependent on $\mathrm{H^+}$.
In order to simulate the redox potential V (in Volts) from the catalyst concentrations $\mathrm{x_3 = [Ru(bpy)_3^{3+}]}$ and $\mathrm{x_{10} = [Ru(bpy)_3^{2+}]}$, the classic Nernst equation is used:

$$\mathrm{V} = V_0 + \frac{\mathrm{RT}}{\mathrm{nF}}\ln\left(\frac{\mathrm{x_3}}{\mathrm{x_{10}}}\right)$$

where

$$R = 8.3144621\ JK^{-1}mol^{-1}$$
$$T = 285.15\ K$$
$$n = 1$$
$$F = 96485.3365\ C\ mol^{-1}$$
$$E_0 = 0.7\ V$$

Section 5: Alphabet chemical representation and dominant pathways in FKN

Extensive simulations with this well-established model, combined with experimentation, were carried out during the design phase to choose appropriate chemical recipes for the input alphabet ensuring distinct and systematic observable dynamic behaviors in the redox potential.

There are 4 subnetworks in FKN[38] that are relevant for understanding the observed redox potential behaviors. Process A is the reduction of bromate $NaBrO_3$ by bromide followed by the bromination of malonic acid, given by $3r_1 + r_2 + r_3 + 3r_8$:

$$BrO_3^- + 2\,Br^- + 3H^+ + 3CH_2(COOH)_2 \rightarrow 3CHBr(COOH)_2 + 3H_2O \quad Process\ A$$

Process G corresponds the autocatalytic production of $HBrO_2$, given by $r_5 + 2r_6$:

$$2[Ru(bpy)_3^{2+}] + BrO_3^- + HBrO_2 + 3H^+ \rightarrow 2[Ru(bpy)_3^{2+}] + 2HBrO_2 + H_2O \quad Process\ G$$

Process B corresponds to the autocatalytic loop (Process G) and the decomposition reaction of $HBrO_2$:

$$2[Ru(bpy)_3^{2+}] + BrO_3^- + HBrO_2 + 3H^+ \rightarrow 2[Ru(bpy)_3^{2+}] + 2HBrO_2 + H_2O \qquad Process\ G$$

$$2HBrO_2 \rightarrow HOBr + BrO_3^- + 2H^+ \qquad Decomposition$$

Process C corresponds to the reduction reactions of the catalyst by malonic and bromomalonic acids, i.e r_9 and r_{10}.

Oscillations occur when the reaction switches between process A and B depending on whether bromide ion concentration is above (process A) or below (process B) a critical value Br^-_{critic}.

Our choice for the chemical representation of the input alphabet is:

- $NaBrO_3$, our chosen "a", enhances dominantly Process B and increases the critical bromide Br^-_{critic} to switch between processes A and B. The more a's the input has, the higher the frequency of the redox oscillations at the onset of oscillations, the smaller the amplitude and the more displaced towards higher redox values (i.e. smaller *D*).
- $CH_3(COOH)_2$, our chosen "b", enhances dominantly Processes A and C and reaction r_{11} thus systematically accelerating the frequency of the redox oscillations but maintaining the amplitude and displacing the oscillations towards lower redox values (i.e. larger *D*).
- NaOH, our chosen "c" decreases the proton H^+ concentration (reaction r_{12}) and consequently slows down Processes A and G, thus systematically decelerating the frequency of redox oscillations, and displacing the oscillations towards lower redox values (i.e. larger *D*).
- $Ru(bpy)_3Cl_2$, our chosen "#", affects dominantly the two redox reactions, Processes G and C, resulting in a systematic deceleration of the frequency of redox oscillations and increased amplitude of oscillations.

Note that each of the symbols does not affect exclusively a single individual reaction in the kinetic mechanism. Instead, each symbol affects simultaneously multiple reactions changing the dominant processes, or pathway, in the reaction. We have a mapping alphabet symbol – dominant pathway (not alphabet symbol – individual reaction), leading to distinct and systematic changes in the redox potential behavior. Such a mapping allows us to inductively argue that the computation works at larger n's that one can test experimentally (due to the interference of produced gas with the monitoring system).

Section 6: Recipe Optimization for the BZ-TM recognizing L_3

The first optimization of the recipe for the language $L_3=\{a^nb^nc^n, n\geq 1\}$ was carried out experimentally with the criteria of maximizing the word length that could be tested experimentally without too much perturbance in the redox potential monitoring system, i.e. by minimizing CO_2 production while ensuring maintaining the oscillatory regime. Such an approach was successful and gave clearly distinct responses in a frequency-distance plot for words in the language from those not belonging to it. However, we still tried to find a more intuitive and straightforward way to interpret the oscillatory response of the chemical TM. We found that by plotting the area $A^{(word)}$ between the fully oxidized redox potential and the oscillations after the end-of-sequence symbol # was added, the machine's answer was "linearized" (see main text Figure 4). This led us to optimizing the recipe so that for words in the language this area would have a constant value independently of word length. Hence, the "Area" measure would work for the chemical TM as the "pH" measure did work for the 1-stack PDA.

Since we have a reasonably realistic model of the chemical kinetics of Belousov-Zhabotinsky we can mathematically optimize the recipe. First, we verified that the mathematical model is representative of the chemistry, and we found out that the model reproduces qualitatively quite well the actual results for $n\geq 2$,

but the model deviates too much for n=1. Therefore, in the optimization we excluded the word *abc.* Then, we studied whether there would be collinearities if we attempted to optimize simultaneously the three elements in the recipe (a,b,c). From this analysis, we concluded that it was best to just optimize one of the recipe aliquots, namely the volume of the bromate aliquot *a*. Next, the optimization problem was defined as follows:

- The optimization variable was chosen to be the step change in concentration of $NaBrO_3$ (M) every time an aliquot of ***a*** is added. If the stock solution is maintained at 2M, then this result is easily transformed into the volume of the aliquot (mL)
- The objective function to be minimized was defined as:

$$F = \sum_i \left(A_i^{(word)} - A_{n=2}^{(word)}\right)^2$$

where $A_i^{(word)}$ is the area for word with n=*i* and $A_{n=2}^{(word)}$ is the area corresponding to aabbcc (e.g. $a^2b^2c^2$).

- The Nelder-Mead nonlinear gradient-free algorithm was chosen as optimization algorithm, since the dynamics of Belousov-Zhabotinsky are highly nonlinear, and discontinuities in the model simulations may take place. Gradient-free methods are slow but safe for this type of problem. The algorithm was implemented using the fminsearch solver in Matlab® (R2015b v. 8.6.0.267246). The solver ode23tb was chosen for solving the system of ODEs (since it is considerably faster than ode23s, and the optimizer requires solving the ODE's many times). The maximum number of iterations was set to 20 and gives a good performance. Simulations were run on a laptop with an Intel® Core™ i7-2620M CPU @2.70 GH processor and Windows 64 Bits Operating System.

Following this procedure, the mathematical optimization results suggested that if the volume of the aliquot of 2M $NaBrO_3$ was increased from 2ml to 2.25 ml, then the area $A^{(word)}$ would become independent of *n* for the words in the language, as illustrated in Figure S2 with the simulation results with the original and the optimized recipe. Finally, the resulting optimized recipe was tested in practice. Since the model is not perfect, the area still showed a very small positive linear dependence with the string length. Dilution effects that are not accounted for in the model explain that the change in area was less in practice than in simulations. However, the action of increasing the volume of sodium bromate was still considered correct, and hence we run the experiments increasing *a* to a value of 2.50 ml, this time the area dependence on string length first increased and then decreased, thus deviating from linearity. A final adjustment of the aliquot volume *a* to 2.40 ml gave the desired results, the area was practically independent of *n* for words in the language. We have thus proven that our method for optimization of the recipe works and gives the correct action change on the optimization variable. Supplementary material SI-4 summarizes the experimental results for both recipe 1 and the mathematically optimized recipe.

One key question is: what is the chemical significance of an area $A^{(Word)}$ that is independent of the string length? For for words in the language, a balance between the reduction and oxidation subnetworks is achieved and maintained. The system is kept at a constant distance from full oxidation because an increment in the oxidation rates is balanced by an increment in the reduction subnetwork. Mathematically, we define area $A^{(Word)}$ as follows:

$$A^{(Word)} = V_{max} \cdot \tau' - \int_{t_\#+30}^{t_\#+\tau} V_{osc}(t)dt$$

where $t_\#$ is the time in reaction coordinates at which the end-of-expression symbol is added, τ' is the time interval between symbols minus 30 seconds (the first 30 seconds are discarded in the integration to allow for fast transients to dissipate), V_{max} is the maximum redox potential (all catalyst in oxidized form) and V_{osc} is the measured redox potential, which can be approximated by the Nernst equation

$$V_{osc} = V_0 + \frac{RT}{nF}\ln\left(\frac{[\mathrm{Ru(bpy)_3^{3+}}]}{[\mathrm{Ru(bpy)_3^{2+}}]}\right),$$

$[\mathrm{Ru(bpy)_3^{2+}}]$ and $[\mathrm{Ru(bpy)_3^{3+}}]$ are respectively the reduced and oxidized form of the catalyst, and can be written in terms of the extent of reaction for the elementary redox reactions in section S4:

$$[\mathrm{Ru(bpy)_3^{3+}}] = [\mathrm{Ru(bpy)_3^{3+}}]_{input} + \xi_6 - 6\xi_9 - 4\xi_{10}$$

$$[\mathrm{Ru(bpy)_3^{2+}}] = [\mathrm{Ru(bpy)_3^{2+}}]_{input} - \xi_6 + 6\xi_9 + 4\xi_{10}$$

$$V_{osc} = V_0 + \frac{RT}{F}\ln\left(\frac{[\mathrm{Ru(bpy)_3^{3+}}]_{input} + \xi_6 - 6\xi_9 - 4\xi_{10}}{[\mathrm{Ru(bpy)_3^{2+}}]_{input} - \xi_6 + 6\xi_9 + 4\xi_{10}}\right)$$

The extents of reaction ξ_i both for the oxidation subnetwork and for the reduction subnetwork increase as the word length is longer, but for this language they increment in a "balanced" way, such that the area is constant.

Finally, the redox potential is related to the Gibbs energy ΔG as follows:

$$\Delta G_{osc} = -n_e F V_{osc}$$

We can thus rewrite the area we have defined above in terms of the Gibbs energy[22] corresponding to full oxidation $\Delta G'$and the redox Gibbs energy ΔG_{osc}:

$$A^{(Word)} = -\frac{1}{n_e F}\left(\Delta G' \cdot \tau' - \int_{t_\# + 30}^{t_\# + \tau} \Delta G_{osc}(t)\ \cdot dt\right)$$

And if the area $A^{(Word)}$ is constant and independent of string length for the words in L_3, so is the integral of ΔG_{osc}, which has important implications for the energetic cost of computation.

Section 7: Supplemental analysis of experimental results for the BZ-TM recognizing L_3

Results for Recipe 1

A comprehensive experimental campaign, comprised of 19 input strings, 5 words in the language L_3 and 14 strings not in the language L_3, was successfully carried out for Recipe 1. The chosen strings not only covered all distinct scenarios leading to a reject (R_1 , R_2, R_3, R_4, R_5) but we also tested "worst-case" strings, i.e. strings not recognized but nearest, both abstractly and chemically, to accepted words

Panel a) of Figure S3 displays the five words recognized as belonging to language L_3 , i.e. accepted strings, that have been tested experimentally, for n=1, 2, 3, 4 and 5, respectively. All these words have in common the following sequence of dynamic trends: the first ***a*** leads to full oxidation of the catalyst (to a maximum redox value), then if there are subsequent ***a***'s the redox value steps down in small steps; otherwise, the first ***b*** leads to onset of oscillations; then, if there are subsequent ***b***'s the frequency is accelerated showing a small step upwards right after the addition of the symbol and an increased positive slope. Otherwise, the first ***c***`s show a deceleration of the frequency, the slope of the frequency might still be positive but with a lower slope. If there are more ***c***´s the frequency is further decelerated, and its slope can become negative. For this recipe, using nonlinear least squares regression we get the following approximate dependence of the frequency on the reactant concentrations):

$$f \sim [BrO_3^-]^{\alpha} \times [MA]^{\beta} \times [NaOH]^{-\gamma}$$

$$f \sim [BrO_3^-]^{0.7434} \times [MA]^{0.1862} \times [NaOH]^{-0.2081}$$

Hence for this recipe and TM operation combination $\alpha > \beta \sim |\gamma| > 0$.

Finally, the end of expression symbol is recognized because it leads to simultaneous decreases in the frequency and the distance of the oscillations. For each recipe, there is a ***nonlinear function*** relating the final frequency $f_\#$ and final distance $D_\#$ at the end of computation $f_\#$ = *f*($D_\#$) which for the recipe 1 used in this campaign adjusts well to the following nonlinear quadratic function:

$$f_\# = -0.0000172\ {D_\#}^2 + 0.007 D_\# - 0.67$$

Where $f_{\#}$ is in Hz, and $f_{\#}$ in mV, and the goodness of fit is R^2=0.9900.

Panel b) of Figure S3 shows examples of strings that are rejected during computation because of wrong order of symbols, before reaching the end-of-expression symbol # and even before oscillations take place:

- Strings that do not begin with ***a*** lack the full oxidation of the catalyst, i.e. if the word start with ***b*** or ***c*** the Ruthenium-based catalyst remains in the reduced state Ru(II) instead of being oxidized to Ru(III), hence the redox value remains at its minimum. See string *bca,* top plot in panel b)*,* and observe that the redox potential upon processing "b" (highlighted in a red square) remains in the reduced value, thus the string is rejected upon processing the first symbol "b".
- Strings that contain the substring ***ac*** lack the stepwise decrease of the redox value that characterizes subsequent ***a***'s. See string *acb,* second plot from the top in panel b)*,* and observe that the redox potential upon processing "c" (highlighted in a red square) remains in the oxidized (maximum) value, thus the string is rejected upon processing the second symbol "c".

Panel b) of Figure S3 shows also examples of strings that are rejected during computation, before reaching the end-of-expression symbol #, but already in the oscillatory regime:

- Strings that contain the substring ***ba*** show a dynamic behavior of the oscillations that is never occurring in words in the language L_3, and that is characterized by a decrease in distance *D* concomitantly with a sharp increase in frequency *f*. It is also characterized by a decrease in amplitude, which is obvious from the redox measurement. See the string *aba,* third plot from the top in panel b)*,* and observe how upon reading the second "a", highlighted in a red square, *D* decreases simultaneously as *f* increases, thus the string is rejected upon reading the symbol in third position "a".
- Strings that contain the substring ***ca*** or ***cb*** show another dynamic behavior of the oscillations that is never occurring in words in the language L_3, and that is characterized by a reacceleration of frequency *f* after the frequency had already began decelerating. Remember that when ***c***'s are read, frequency decelerates, so that a reacceleration means that the current symbol cannot be a subsequent ***c*** but an ***a*** or ***b***. See string *aabccb*, bottom plot in panel b)*,* and observe how upon reading the "b" in substring "cb", highlighted in a red square, *f* changes trend from decreasing to increasing again, thus the string is rejected upon processing the symbol in sixth position "b".

The input strings for Recipe 1 that were rejected upon processing the end-of-expression symbol #, are shown in Figure S4. Panel a) shows four examples of strings that are rejected because the number of ***a***'s exceeds that of ***b***'s and ***c***'s. If we compare *aabc, aaabbcc, aaaabbbccc* with their nearest word in the language, i.e. n=1, 2 and 3 respectively, we observe in the three cases that the values at the end of computation are displaced towards lower values of distance and larger values of frequency. Hence, in the final frequency final distance plot, the words that have excess *a´*s are displaced northwest with respect to the function fulfilled by words in the language. Panel a) also shows one string that is rejected once the end of expression symbol is reached because the number of ***a***'s and ***b***'s exceed that of ***c***'s, *aabb*.

Panel b) of Figure S4 shows two examples of strings that are rejected once the end of expression symbol is reached because the number of ***b***'s exceeds that of ***a***'s and ***c***'s: *abbc* and *aabbbcc*. If we compare these with the nearest word in the language, e.g. for *abbc* the nearest word in the language is *abc*, we observe in the two cases that the values at the end of computation are displaced toward larger values of distance and larger values of frequency than its nearest word in the language. Hence, in the final frequency final distance plot, the words that have excess ***b***´s are displaced northeast with respect to the function fulfilled by words in the language. Panel b) of Figure S4 also shows two examples of strings that are rejected once the end of expression symbol is reached because the number of ***c***'s exceeds that of ***a***'s and ***b***'s: *aaabbbcccc* and *aaaabbbbccccc*. If we compare them with the nearest word in the language we observe in the two cases that the values at the end of computation are displaced toward larger values of distance and smaller values of frequency than its nearest word in the language. Hence, in the final frequency final distance plot, the words that have excess ***c***´s are displaced southeast with respect to the function fulfilled by words in the language.

The experiments have shown that our realization of the BZ TM gives clearly distinct responses for strings that belong to the language $L_3 = \{a^n b^n c^n\}$ than for strings that are not in the language. Moreover, each type

of reject in the abstract machine has a differentiated type of chemical response in the redox potential in the BZ machine. Indeed, all rejects that occur in the abstract machine during computation have their distinct chemical counterpart in the BZ machine, as it was expected.

Results for Recipe 2

Another extensive experimental campaign was successfully carried out using the mathematically optimized Recipe 2 (more concentrated $NaBrO_3$ aliquot). The campaign was comprised of 12 input strings, 5 of them were words in the language L_3 and 7 were rejected upon reaching the end-of-expression symbol #. Panel a) of Figure S5 shows the five accepted words, for n=1, 2, 3, 4 and 5, arranged from top to bottom, respectively. All these words fulfill the same sequence of dynamic trends as described for Recipe 1. Panel b) of Figure S5 shows the dynamic trends for 5 of the 7 tested rejected input strings, and again the results are qualitatively consistent with their counterparts for recipe 1. The rejected input strings are clustered in the same relative positions (e.g. NW for excess ***a***'s, NE for excess ***b***'s, SE for excess ***c***'s) with respect to the canonical language L_3. For most of the input strings, accepted or rejected, the result for recipe 2 is displaced toward a higher frequency and a lower distance that the result for its counterpart using recipe 1.

For Recipe 2 the ***nonlinear function*** relating the final frequency $f_\#$ and final distance $D_\#$ at the end of computation $f_\# = f(D_\#)$ adjusts well to the following nonlinear quadratic function:

$$f_\# = -0.0000395\ {D_\#}^2 + 0.015 D_\# - 1.43$$

where $f_\#$ is in Hz, and $D_\#$ in mV, and the goodness of fit is R^2=0.9930.