

Computing with chemical reaction networks: a tutorial

Peer-reviewed author version

BRIJDER, Robert (2019) Computing with chemical reaction networks: a tutorial. In: Natural Computing, 18 (1), p. 119-137.

DOI: 10.1007/s11047-018-9723-9

Handle: <http://hdl.handle.net/1942/30009>

Computing with Chemical Reaction Networks: A Tutorial

Robert Brijder

Hasselt University, Belgium

Abstract

Chemical reaction networks (CRNs) model the behavior of chemical reactions in well-mixed solutions and they can be designed to perform computations. In this tutorial we give an overview of various computational models for CRNs. Moreover, we discuss a method to implement arbitrary (abstract) CRNs in a test tube using DNA. Finally, we discuss relationships between CRNs and other models of computation.

1 Introduction

Chemical reaction networks are a fundamental model of chemical reactions in well-mixed solutions. A *chemical reaction network* (CRN) is, roughly, a finite set of chemical reactions like $X + Y \rightarrow 2Y + Z$, where X , Y , and Z are “abstract” molecular species, i.e., these species are not tied to any chemical implementation. CRN theory [31], which studies the dynamic behavior of CRNs, is a mature research field that is traditionally focused on networks of chemical reactions occurring in nature. Recently, it has been shown that carefully designed CRNs are able to compute [46, 61] — such computations can, e.g., take place in a test tube. There is now a rapidly growing body of literature devoted to computational models for (abstract) CRNs. Decoupling reactions from chemical implementations introduces a higher level of abstraction, where CRNs become a high-level *programming language*. Using a (semi-)automated method, arbitrary abstract CRNs can then be *compiled* to chemical implementations [62, 23, 7].

In this tutorial we give a gentle overview of the popular computational models for CRNs, a possible chemical implementation for arbitrary CRNs, and an overview of related models of computation.

First we define the notion of a CRN and the way it operates on a discrete state space (Section 2). A discrete state describes the counts of the species of a CRN. In Subsections 3.1 and 3.2 we turn to a model of computation for such discrete CRNs inspired by the notion of population protocols from the research field of distributed computing [3]. Here a CRN computes by either accepting or rejecting an input state, much like a finite state automaton or a Turing machine

accepts or rejects arbitrary input strings. Equivalently, such a CRN can be seen as recognizing a Boolean-valued function on its set of input states. This computational model is then extended in Subsection 3.3 to the computation of more general functions than Boolean-valued functions [21]. Next we study CRNs that never need “slow” reactions to perform their computations (such slow reactions are called speed faults) [20].

In Subsection 4.1, we augment discrete CRNs with stochastics to obtain stochastic CRNs. In this way, a stochastic CRN behaves as a continuous-time Markov chain on the discrete state space of the CRN. The probability for a reaction to take place here depends on (1) the molecular counts of the reactants, (2) the volume of the solution (e.g., the test tube), and (3) the rate constant (a value that depends on the reaction). With the notion of a stochastic CRN in place, we show in Subsection 4.2 that stochastic CRNs can simulate Turing machines with an arbitrary small positive probability of error [61]. In Subsection 4.3 we turn to the natural problem of leader election [4, 10, 29] and in Subsection 4.4 we show that probability distributions can be computed using CRNs [16].

Instead of considering discrete state spaces, where a discrete state describes the *counts* (which are nonnegative integers) of the species of a CRN, one can also consider a continuous state space, where a continuous state describes the *concentrations* (which are nonnegative real numbers) of the species. In fact, CRN theory has traditionally focused largely on CRNs with continuous state spaces. However, such continuous CRNs only form a faithful approximation of reality in environments where the molecule counts are high (and stay high) [44]. The standard mass-action continuous CRN model is given in Subsection 5.1, including a discussion of the computational mode of operation for this CRN model [30], and the continuous CRN model from [22] is studied from a computational point of view in Subsection 5.2.

In Section 6 we recall from [62] that arbitrary CRNs can be implemented in the wetlab using DNA as a substrate, in this way, justifying the high level of abstraction that was taken in the previous sections. We recall in Section 7 that CRNs are closely related to Petri nets [53, 56], vector addition systems [43], and population protocols [3]. Petri nets and vector addition systems are the most studied models of concurrency and population protocols form a popular model for distributed computing. We end with a discussion.

2 Chemical reaction networks

In this section we recall the notion of a chemical reaction network (CRN), which is roughly a set of reactions. We consider a level of abstraction that is higher than that of concrete chemical reactions. So, rather than considering “concrete” chemical reactions such as $\text{NaHCO}_3 + \text{HCl} \rightarrow \text{H}_2\text{O} + \text{NaCl} + \text{CO}_2$, we abstract from the level of molecular species, like NaHCO_3 and HCl , and instead consider abstract species, usually denoted by capital letters like A and B , and reactions like $2A + B \rightarrow 3A$. Thus we do not worry about the chemical implementations

of species A and B . This higher level of abstraction is justified in Section 6, where it is recalled that *any* (abstract) CRN is implementable in the wetlab using DNA as a substrate.

Let \mathbb{N} be the set of nonnegative integers. Let Λ be a finite set. The set of vectors over \mathbb{N} indexed by Λ (i.e., the set of functions $\varphi : \Lambda \rightarrow \mathbb{N}$) is denoted by \mathbb{N}^Λ . A vector $\mathbf{v} \in \mathbb{N}^\Lambda$ can be viewed as a multiset with Λ as the underlying set. For $\mathbf{x} \in \mathbb{N}^\Lambda$, we denote the cardinality of the multiset \mathbf{x} by $\|\mathbf{x}\| = \sum_{i \in \Lambda} \mathbf{x}(i)$. We denote the restriction of \mathbf{x} to $\Sigma \subseteq \Lambda$ by $\mathbf{x}|_\Sigma$. For $\mathbf{x}, \mathbf{y} \in \mathbb{N}^\Lambda$ we write $\mathbf{x} \leq \mathbf{y}$ if and only if $\mathbf{x}(i) \leq \mathbf{y}(i)$ for all $i \in \Lambda$.

A *reaction* α over Λ is a tuple (\mathbf{r}, \mathbf{p}) with $\mathbf{r}, \mathbf{p} \in \mathbb{N}^\Lambda$; \mathbf{r} and \mathbf{p} are called the *reactants* and *products* of α , respectively. A reaction is commonly written additively, where, e.g., $A + 2B \rightarrow B + C$ denotes the reaction (\mathbf{r}, \mathbf{p}) over $\Lambda = \{A, B, C\}$, where $\mathbf{r}(A) = 1$, $\mathbf{r}(B) = 2$, $\mathbf{r}(C) = 0$, $\mathbf{p}(A) = 0$, $\mathbf{p}(B) = 1$, and $\mathbf{p}(C) = 1$. Reaction α is called *unimolecular* if $\|\mathbf{r}\| = 1$, and *bimolecular* if $\|\mathbf{r}\| = 2$. Since it is very rare in nature for three or more molecules to simultaneously collide, almost all elementary reactions (that is, reactions that cannot be decomposed into multiple reactions) in nature are unimolecular and bimolecular reactions. Reaction α is called *mute* if $\mathbf{r} = \mathbf{p}$.

We now define the central notion of a chemical reaction network.

Definition 2.1. A chemical reaction network (CRN) is an ordered pair $\mathcal{N} = (\Lambda, R)$ with Λ a finite set and R a finite set of reactions over Λ .

The elements of Λ are called the *species* of \mathcal{N} . Also, the sets of species and reactions of a CRN are denoted by $\Lambda(\mathcal{N})$ and $R(\mathcal{N})$, or, if the CRN under consideration is clear, simply by Λ and R , respectively.

Example 2.2. Let $\mathcal{N} = (\Lambda, R)$ with $\Lambda = \{A, B, C\}$ and $R = \{3A \rightarrow 2B, B + C \rightarrow A, C \rightarrow B, B \rightarrow C\}$. Then \mathcal{N} is a CRN having three species and four reactions. One reaction of \mathcal{N} is bimolecular ($B + C \rightarrow A$) and two are unimolecular ($C \rightarrow B$ and $B \rightarrow C$).

The elements of \mathbb{N}^Λ are called the (discrete) *states* of \mathcal{N} (also called configurations in the literature), and they describe the counts of each of the molecular species of \mathcal{N} in some well-mixed solution (such as a well-mixed test tube). Viewing \mathbf{c} as a multiset, each element of \mathbf{c} is called a *molecule*. So, \mathbf{c} has $\|\mathbf{c}\|$ molecules. A molecule of species S is sometimes called a S -molecule for short. Just as reactions, we often write states additively (assuming the underlying species set Λ is clear from the context). If $S \in \Lambda$ is some species and \mathbf{c} a state then the number $\mathbf{c}(S)$ of S -molecules in \mathbf{c} is sometimes denoted by $\#_{\mathbf{c}} S$ or simply $\#S$ if \mathbf{c} is clear from the context.

As a consequence of the well-mixedness assumption, if all reactants of some reaction $\alpha = (\mathbf{r}, \mathbf{p})$ are available in sufficient quantity (i.e., $\mathbf{r} \leq \mathbf{c}$), then α can take place. This is formalized as follows.

For a state $\mathbf{c} \in \mathbb{N}^\Lambda$ and a reaction α over Λ , we say that $\alpha = (\mathbf{r}, \mathbf{p})$ is *applicable* to \mathbf{c} if $\mathbf{r} \leq \mathbf{c}$. If α is applicable to \mathbf{c} , then the *result* of applying α to \mathbf{c} , denoted by $\alpha(\mathbf{c})$, is $\mathbf{c}' = \mathbf{c} - \mathbf{r} + \mathbf{p}$. In this case we also write $\mathbf{c} \Rightarrow_\alpha \mathbf{c}'$. Note

that \mathbf{c}' is a state, i.e., $\mathbf{c}' \in \mathbb{N}^\Lambda$. We also write $\mathbf{c} \Rightarrow_{\mathcal{N}} \mathbf{c}'$ to denote that $\mathbf{c} \Rightarrow_{\alpha} \mathbf{c}'$ for some reaction α of \mathcal{N} . The transitive and reflexive closure of $\Rightarrow_{\mathcal{N}}$ is denoted by $\Rightarrow_{\mathcal{N}}^*$. If $\mathbf{c} \Rightarrow_{\mathcal{N}}^* \mathbf{c}'$, then we say \mathbf{c}' is *reachable* from \mathbf{c} in \mathcal{N} .

Example 2.3. Consider again the CRN \mathcal{N} of Example 2.2. Consider the state $\mathbf{c} = A + 2C$. Then only the reaction $C \rightarrow B$ of \mathcal{N} is applicable to \mathbf{c} . We have $\mathbf{c} \Rightarrow_{\mathcal{N}} \mathbf{c}'$ where $\mathbf{c}' = A + B + C$, in other words \mathbf{c}' is the result of applying $C \rightarrow B$ to \mathbf{c} . Three reactions of \mathcal{N} are applicable to \mathbf{c}' . For example, we have $\mathbf{c}' \Rightarrow_{\mathcal{N}} 2A$. In state $2A$ no reactions of \mathcal{N} are applicable. We observe that, e.g., $2A$ is reachable from \mathbf{c} .

Remark 2.4. We remark that a reaction α is usually defined as a triple, consisting also of a positive real number k_{α} called the rate constant of α which determines the likelihood of the reaction to take place in the current state (assuming it is applicable). Since this section and the next section only deals with reachability (i.e., whether it is possible to reach one state from another), we postpone considering rate constants until Section 4.

For $\mathbf{c} \in \mathbb{N}^\Lambda$, we define $\text{pre}_{\mathcal{N}}(\mathbf{c}) = \{\mathbf{c}' \in \mathbb{N}^\Lambda \mid \mathbf{c}' \Rightarrow_{\mathcal{N}}^* \mathbf{c}\}$ and $\text{post}_{\mathcal{N}}(\mathbf{c}) = \{\mathbf{c}' \in \mathbb{N}^\Lambda \mid \mathbf{c} \Rightarrow_{\mathcal{N}}^* \mathbf{c}'\}$. So, $\text{post}_{\mathcal{N}}(\mathbf{c})$ contains all states that can be reached from \mathbf{c} (including \mathbf{c} itself), and $\text{pre}_{\mathcal{N}}(\mathbf{c})$ contains all states that can reach \mathbf{c} (including \mathbf{c} itself). A state $\mathbf{c} \in \mathbb{N}^\Lambda$ is called *terminal* in \mathcal{N} if $\text{post}_{\mathcal{N}}(\mathbf{c}) = \{\mathbf{c}\}$. In other words, a state is terminal if no non-mute reaction of \mathcal{N} is applicable to \mathbf{c} .

If \mathcal{N} is clear from the context, then we often omit the subscripts of $\Rightarrow_{\mathcal{N}}$, $\Rightarrow_{\mathcal{N}}^*$, $\text{pre}_{\mathcal{N}}$ and $\text{post}_{\mathcal{N}}$.

We extend $\text{pre}(\mathbf{c})$ and $\text{post}(\mathbf{c})$ to sets $X \subseteq \mathbb{N}^\Lambda$ of states in the natural way: $\text{pre}(X) := \bigcup_{\mathbf{c} \in X} \text{pre}(\mathbf{c})$, and $\text{post}(X) := \bigcup_{\mathbf{c} \in X} \text{post}(\mathbf{c})$.

We remark here that the notion of a CRN is similar to some notions from other research domains, see Section 7 for details. Therefore, the results presented here can (often) be straightforwardly carried over to these domains.

3 Computing with discrete chemical reaction networks

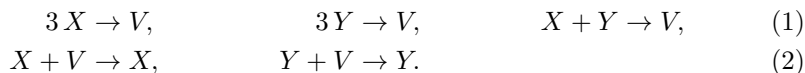
For a significant part of the rest of the paper we recall several models of computing with CRNs from the literature and discuss some of their key results. The computational CRN models this only concern reachability of states, and so their results are independent of stochastics (i.e., how likely a certain state is reached). CRNs augmented with stochastics are discussed in Section 4.

For didactical reasons we do not discuss the computational CRN models in chronological order, but instead we first consider the elementary computational model introduced in [21], which is in turn inspired by (and very similar to) the computational model of Population Protocols [3] (see Subsection 7.3 for a comparison between Population Protocols and CRNs).

3.1 Haltingly deciding chemical reaction deciders

Suppose we are given a state with an unknown number of molecules of species X and Y and we want to decide whether or not $\#X$ is equal to $\#Y$ modulo 3. Is there a CRN that can perform this computation? More specifically, is there a CRN which eventually (by keeping applying reactions) halts on every possible input such that merely the presence of certain molecules in the halting state indicates whether the answer to the decision problem is yes or no? The next example (taken from [14]) shows that the above given modulo problem $\#X \stackrel{?}{\equiv} \#Y \pmod{3}$ can be decided by a CRN.

Example 3.1. Consider the CRN $\mathcal{N} = (\Lambda, R)$ with $\Lambda = \{X, Y, V\}$ and R consisting of the following reactions



First notice that all reactions preserve whether or not $\#X \equiv \#Y \pmod{3}$. The reactions of (1) all reduce the number of X or Y molecules, while the remaining reactions (of (2)) do not influence the X and Y molecules. So, eventually (that is, when reactions continue to take place), we reach a state where none of the reactions of (1) can take place anymore. The last reaction of (1) that took place introduced a V -molecule. Now, if $\#X \equiv \#Y \pmod{3}$, then no X and Y molecules are present anymore at this point and so the CRN has halted with only some V -molecules. If $\#X \not\equiv \#Y \pmod{3}$, then some X - or Y -molecules remain and these eat all the V -molecules that are present by the reactions of (2). So, in this case the CRN eventually halts with only some X - or Y -molecules.

Consequently, eventually the CRN halts and the presence of V -molecules in the terminal state indicate that $\#X \equiv \#Y \pmod{3}$ holds, while the presence of X - or Y -molecules in the terminal state indicate that $\#X \equiv \#Y \pmod{3}$ does not hold. We then say that V is a yes voter (or 1-voter) and X and Y are no voters (or 0-voters). Note however that this computation does not work in the corner case where the initial state has no molecules, since in this case no V -molecule is ever produced.

Inspired by Example 3.1 we now formalize the above illustrated model of computation for CRNs. First we define the notion of chemical reaction decider which is roughly a CRN augmented with three distinguished sets of species: one to define the input states, one to define the no voters, and one to define the yes voters.

Definition 3.2. A chemical reaction decider (CRD) is a 4-tuple $\mathcal{D} = (\mathcal{N}, \Sigma, \Lambda_0, \Lambda_1)$, where \mathcal{N} is a CRN, $\Sigma, \Lambda_0, \Lambda_1 \subseteq \Lambda(\mathcal{N})$, and $\Lambda_0 \cap \Lambda_1 = \emptyset$.

The elements of Σ , Λ_0 , and Λ_1 are called the *input species*, *0-voters*, and *1-voters*, respectively. The elements of $\mathbb{N}^\Sigma \setminus \{\mathbf{0}_\Sigma\}$ are called the *input states*, where by $\mathbf{0}_\Sigma$ we denote the zero vector with index set Σ . If the index set is clear from the context we just write $\mathbf{0}$ instead of $\mathbf{0}_\Sigma$. For $b \in \{0, 1\}$, let

$\mathcal{L}_b = \{\mathbf{c} \in \mathbb{N}^\Lambda \mid c_{\Lambda_b} \neq \mathbf{0}\}$ be the set of states that have at least one b -voter. We say that \mathbf{c} has *output* $b \in \{0, 1\}$ if $\mathbf{c} \in \mathcal{L}_b \setminus \mathcal{L}_{1-b}$. In other words, \mathbf{c} has output b when it contains b -voter molecules, but no $(1 - b)$ -voter molecules.

Let \mathcal{T} be the set of terminal states of \mathcal{N} and let, for $b \in \{0, 1\}$, $\mathcal{T}_b = \mathcal{T} \cap (\mathcal{L}_b \setminus \mathcal{L}_{1-b})$ be the set of terminal states of \mathcal{N} with output b . We say that $\mathbf{c} \in \mathbb{N}^\Lambda$ is *output- b halting* if $\text{post}(\mathbf{c}) \subseteq \text{pre}(\mathcal{T}_b)$. In other words, \mathbf{c} is output- b halting if every state reachable from \mathbf{c} (including \mathbf{c} itself) can reach an output- b terminal state. Note that a state cannot be both output-0 halting and output-1 halting.

Remark 3.3. *It is worthwhile to note that the definition of output- b halting is often sloppily interpreted as saying that starting from an output- b halting state we eventually reach an output- b terminal state. This is incorrect in general since we may, e.g., have a output- b halting (but not terminal) state \mathbf{c} such that $\mathbf{c} \Rightarrow^+ \mathbf{c}$ (where \Rightarrow^+ denotes the transitive closure of \Rightarrow), and so the current state may indefinitely be in a loop without reaching an output- b terminal state (we correctly used “eventually” in Example 3.1 since loops cannot appear there). To ensure eventually reaching an output- b terminal state, it is possible to additionally assume some notion of fairness [21]. One such notion of fairness is implicit for stochastic chemical reaction networks (see Section 4) that have only a finite number of states reachable from any given state [25].*

For $\mathbf{c} \in \mathbb{N}^\Sigma$, let $\imath(\mathbf{c}) \in \mathbb{N}^\Lambda$ be the vector obtained from \mathbf{c} by padding zeros for the entries indexed by $\Lambda \setminus \Sigma$, i.e., $\imath(\mathbf{c})|_\Sigma = \mathbf{c}$ and $\imath(\mathbf{c})|_{\Lambda \setminus \Sigma} = \mathbf{0}$.

We now define a key notion.

Definition 3.4. *We say that a CRD \mathcal{D} is haltingly deciding if, for each input state \mathbf{c} of \mathcal{D} , $\imath(\mathbf{c})$ is output- b halting for some $b \in \{0, 1\}$.*

Thus \mathcal{D} is haltingly deciding if for each input state \mathbf{c} , there is a $b \in \{0, 1\}$ such that during the computation a terminal state with output b is always reachable. So, starting from \mathbf{c} you can never go “wrong” since a terminal state with output b always remains reachable.

If \mathcal{D} is haltingly deciding, then we say that \mathcal{D} (*haltingly*) *recognizes* the set $\{\mathbf{c} \in \mathbb{N}^\Sigma \setminus \{\mathbf{0}\} \mid \imath(\mathbf{c}) \text{ is output-1 halting}\}$. If a haltingly deciding \mathcal{D} recognizes $X \subseteq \mathbb{N}^\Sigma \setminus \{\mathbf{0}\}$, then we also say that \mathcal{D} *decides* the predicate, i.e., Boolean-valued function, $\varphi : \mathbb{N}^\Sigma \setminus \{\mathbf{0}\} \rightarrow \{0, 1\}$ where $\varphi(x)$ holds (i.e., $\varphi(x) = 1$) if and only if $x \in X$.

Example 3.5. *Consider again the CRN \mathcal{N} from Example 3.1. With the notions and terminology in place we can now formalize the behavior of \mathcal{N} as a CRD $\mathcal{D} = (\mathcal{N}, \Sigma, \Lambda_0, \Lambda_1)$, where $\Sigma = \Lambda_0 = \{X, Y\}$ and $\Lambda_1 = \{V\}$. From the observations we made in Example 3.1 we conclude that \mathcal{D} haltingly recognizes the set $\{\mathbf{c} \in \mathbb{N}^\Sigma \setminus \{\mathbf{0}\} \mid \mathbf{c}(X) \equiv \mathbf{c}(Y) \pmod{3}\}$.*

Remark 3.6. *Note that, when a CRD halts on a given input, the CRD does not give a “signal” that it has halted. In other words, an observer of a computation of the CRD does not know whether or not the output of the computation is final*

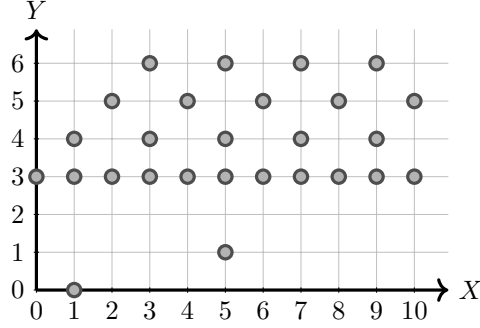


Figure 1: A semilinear set.

unless it has determined somehow that the computation has terminated (i.e., that no non-mute reaction can take place in the current state). One could imagine an alternative mode of operation for CRDs in which the presence of at least one molecule of some distinguished species signals that the output is final.

It is natural to ask which sets can be haltingly recognized by CRDs. We say that $X \subseteq \mathbb{N}^\Sigma$ is *linear* if there is a finite set $S \subseteq \mathbb{N}^\Sigma$ and a $\mathbf{d} \in \mathbb{N}^\Sigma$ such that $X = \{\mathbf{d} + \sum_{\mathbf{v} \in S} n_{\mathbf{v}} \mathbf{v} \mid n_{\mathbf{v}} \in \mathbb{N} \text{ for all } \mathbf{v} \in S\}$. We say that $X \subseteq \mathbb{N}^\Sigma$ is *semilinear* if X is the union of a finite number of linear sets. We remark that semilinear sets are precisely the sets definable in Presburger arithmetic, which is the first-order theory of natural numbers with addition [34].

Example 3.7. Let $\Sigma = \{X, Y\}$. Consider the following four linear sets: L_1 is defined by $S_1 = \{X + Y, 2X\}$ and $\mathbf{d}_1 = 3Y$, L_2 is defined by $S_2 = \emptyset$ and $\mathbf{d}_2 = X$, L_3 is defined by $S_3 = \emptyset$ and $\mathbf{d}_3 = 5X + Y$, and L_4 is defined by $S_4 = \{X\}$ and $\mathbf{d}_4 = 3Y$. The semilinear set $L_1 \cup L_2 \cup L_3 \cup L_4$ is depicted in Figure 1.

The following result combines results from [3] and [5]. The if direction has been shown in the proof of the main result of [3] (see [12] for some details concerning the halting claim), and the only-if direction is a special case of the main result of [5].

Theorem 3.8 ([3, 5]). Let $X \subseteq \mathbb{N}^\Sigma \setminus \{\mathbf{0}\}$. Then X is recognized by a haltingly deciding CRD if and only if X is semilinear.

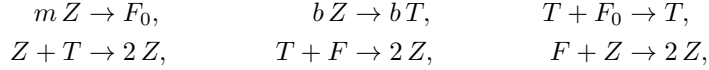
Moreover, this result also holds if we restrict to haltingly deciding CRDs \mathcal{D} such that (1) \mathcal{D} has only bimolecular reactions and (2) every species of \mathcal{D} is either a 0-voter or a 1-voter.

It is well known (from vector addition system theory) that for a state \mathbf{c} and a CRN \mathcal{N} , $\text{pre}_{\mathcal{N}}(\mathbf{c})$ and $\text{post}_{\mathcal{N}}(\mathbf{c})$ are not necessarily semilinear [36]. This makes Theorem 3.8 rather surprising.

It is well known (see, e.g., [34]) that semilinear sets are exactly the sets that are obtained by finite unions, intersections, and complementations of sets which are either of the form $X_{\mathbf{a},b} = \{\mathbf{x} \in \mathbb{N}^\Sigma \mid \mathbf{a} \cdot \mathbf{x} \leq b\}$ or of the form $X_{\mathbf{a},b,m} = \{\mathbf{x} \in \mathbb{N}^\Sigma \mid \mathbf{a} \cdot \mathbf{x} \equiv b \pmod{m}\}$, where $\mathbf{a} \in \mathbb{Z}^\Sigma$, $b \in \mathbb{Z}$, and $m \in \mathbb{N} \setminus \{0, 1\}$ are constants and \cdot denotes the dot product. The proof of the if direction in [3] shows that (1) there are CRDs that compute $X_{\mathbf{a},b}$ and $X_{\mathbf{a},b,m}$ and (2) if CRDs \mathcal{D}_1 and \mathcal{D}_2 compute the sets X_1 and X_2 , respectively, then there are CRDs that compute $X_1 \cup X_2$, $X_1 \cap X_2$ and $\mathbb{N}^\Sigma \setminus X_1$.

As an example we illustrate why $X_{\mathbf{a},b,m}$ is semilinear.

Example 3.9. First we show that the predicate $\#Z \stackrel{?}{\equiv} b \pmod{m}$, where b and m are nonnegative integers with $b < m$, can be haltingly decided by a CRD. Consider the CRD $\mathcal{D} = (\mathcal{N}, \Sigma, \Lambda_0, \Lambda_1)$ with $\mathcal{N} = (\Lambda, R)$, $\Lambda = \{Z, T, F, F_0\}$, $\Lambda_0 = \{F_0, F\}$, $\Lambda_1 = \{T\}$ and R consisting of the following reactions



and $kZ \rightarrow kF$ for $k \in \{1, \dots, m-1\} \setminus \{b\}$. Note that each of these reactions preserve the value $\#Z + \#F + \#T \pmod{m}$. Because of the reaction $mZ \rightarrow F_0$ it is always possible to reach a state where $\#Z < m$. In fact, by additionally using the three reactions with $2Z$ as the products, it is always possible to reach a state where $x = \#Z + \#F + \#T < m$. If $x \neq 0$, then one easily verifies that the only way to halt is when the three reactions with $2Z$ as the products take place until $\#F + \#T = 0$, and then followed by either (1) reaction $bZ \rightarrow bT$ taking place (when $x = b$), or (2) $kZ \rightarrow kF$ taking place (when $x = k$). In case (2) or when $x = 0$, the CRD has halted with only 0-voters left, and in case (1) the CRN halts with only T -molecules left once $T + F_0 \rightarrow T$ has taken place repeatedly until all F_0 -molecules are gone.

Now, more elaborate examples like $2\#X_1 - \#X_2 \stackrel{?}{\equiv} b \pmod{m}$ can be reduced to the problem $\#Z \stackrel{?}{\equiv} b \pmod{m}$ by extending the above \mathcal{N} with the reactions $X_1 \rightarrow 2Z$ and $X_2 \rightarrow (m-1)Z$ (the latter because $-1 \equiv m-1 \pmod{m}$).

3.2 Stably deciding chemical reaction deciders and other modes of operation

We now present a natural generalization of the notion of haltingly deciding CRDs. Instead of requiring for each input the existence of some $b \in \{0, 1\}$ such that during the computation a terminal state with output b is always reachable, we now merely require that it is always possible to reach a state \mathbf{c} such that any state reachable from \mathbf{c} (including itself) has output b . So, even though non-mute reactions may still take place at \mathbf{c} , any state reachable from \mathbf{c} has the same output b . State \mathbf{c} is then called output- b stable.

More precisely, let $b \in \{0, 1\}$. We say that $\mathbf{c} \in \mathbb{N}^\Lambda$ is *output- b stable* if every $\mathbf{c}' \in \text{post}(\mathbf{c})$ has output b . Let \mathcal{S}_b be the set of output- b stable states. Note that any output- b halting state is output- b stable, i.e., $\mathcal{T}_b \subseteq \mathcal{S}_b$.

Similar as for \mathcal{T}_b , we say that $\mathbf{c} \in \mathbb{N}^\Lambda$ is *output- b stabilizing* (for $b \in \{0, 1\}$) if $\text{post}(c) \subseteq \text{pre}(\mathcal{S}_b)$. Note that a state cannot be both output-0 stabilizing and output-1 stabilizing.

Definition 3.10. *We say that a CRD \mathcal{D} is stably deciding if for each input state \mathbf{c} of \mathcal{D} , $\iota(\mathbf{c})$ is output- b stabilizing for some $b \in \{0, 1\}$.*

If \mathcal{D} is stably deciding, then we say that \mathcal{D} (stably) *recognizes* the set $\{\mathbf{c} \in \mathbb{N}^\Sigma \setminus \{\mathbf{0}\} \mid \iota(\mathbf{c}) \text{ is output-1 stabilizing}\}$.

While it is computationally easy to determine if a state is terminal (one just has to verify whether a non-mute reaction can take place in the given state), it does not seem to be computationally easy to determine if a state is output- b stable for some $b \in \{0, 1\}$ (although it is known to be decidable [12]). However, restricting to the class of CRDs where $\|\mathbf{r}\| = \|\mathbf{p}\| = 2$ for all reactions $\alpha = (\mathbf{r}, \mathbf{p})$, output stability can be shown efficiently — especially, due to a preprocessing step, when a large set of states need to be checked for output stability [12].

The main result of [5] shows that a very general class of CRDs, which in particular includes the stably deciding CRDs, can only compute semilinear sets. By Theorem 3.8, we therefore observe that stably deciding CRDs and haltingly deciding CRDs compute the same family of sets, namely the family of semilinear sets that do not contain the zero vector.

Theorem 3.11 ([3, 5]). *Let $X \subseteq \mathbb{N}^\Sigma$. Then X is recognized by a stably deciding CRD if and only if X is recognized by a haltingly deciding CRD.*

Note that the above definitions of the output of a state are based on consensus: states with both yes and no voters do not have a defined output. One can consider a democratic mode of operation based on majority voting. Also, one can consider a mode of operation without 0-voters (here the existence or absence of 1-voters determines the output). These and other modes of operation have been shown to also compute exactly all semilinear sets not containing the zero vector, see [14].

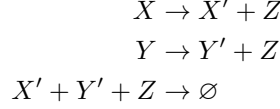
3.3 Computing functions

In the previous subsection we considered a way of computing predicates $\varphi : \mathbb{N}^\Sigma \setminus \{\mathbf{0}\} \rightarrow \{0, 1\}$ using CRNs. Following [21], we now consider the computation of functions of the form $\varphi : \mathbb{N}^\Sigma \rightarrow \mathbb{N}^\Gamma$ using CRNs.

Example 3.12. *Consider the function \min that computes the minimum $\min(x, y)$ of two nonnegative integers x and y . For $\Sigma = \{X, Y\}$ and $\Gamma = \{Z\}$, the function $\min : \mathbb{N}^\Sigma \rightarrow \mathbb{N}^\Gamma$ can be easily seen to be computed through the reaction $X + Y \rightarrow Z$. Indeed, starting from an initial state \mathbf{i} consisting of only X and Y molecules, the CRN eventually halts in a state \mathbf{c} where $\#_{\mathbf{c}}Z = \min(\#_{\mathbf{i}}X, \#_{\mathbf{i}}Y)$.*

Computing the max function turns out to be slightly more involved.

Example 3.13. Consider the function \max that computes the maximum $\max(x, y)$ of two nonnegative integers x and y . For $\Sigma = \{X, Y\}$ and $\Gamma = \{Z\}$, the function $\max : \mathbb{N}^\Sigma \rightarrow \mathbb{N}^\Gamma$ can be computed using the auxiliary species X' and Y' and reactions



Notice that if the third reaction is absent, then, starting from an initial state \mathbf{i} consisting of only X and Y molecules, the CRN eventually halts in a state \mathbf{c} where $\#_{\mathbf{c}}Z = \#_{\mathbf{i}}X + \#_{\mathbf{i}}Y$. Observe that the third reaction consumes exactly $\min(\#_{\mathbf{i}}X, \#_{\mathbf{i}}Y)$ Z -molecules. So, the whole CRN eventually halts in a state \mathbf{c} where $\#_{\mathbf{c}}Z = \#_{\mathbf{i}}X + \#_{\mathbf{i}}Y - \min(\#_{\mathbf{i}}X, \#_{\mathbf{i}}Y) = \max(\#_{\mathbf{i}}X, \#_{\mathbf{i}}Y)$. So, this CRN indeed computes the \max function.

In analogy with the chemical reaction decider we define the chemical reaction computer [21, 28].

Definition 3.14. A chemical reaction computer (CRC) is 3-tuple $\mathcal{C} = (\mathcal{N}, \Sigma, \Gamma)$, where \mathcal{N} is a CRN and $\Sigma, \Gamma \subseteq \Lambda(\mathcal{N})$ are disjoint.

The elements of Σ and Γ are called the *input species* and *output species*, respectively. Similar as for CRDs, the elements of \mathbb{N}^Σ are called the *input states*. We say that a state \mathbf{c} has *output* $\mathbf{c}|_\Gamma$.

We say that $\mathbf{c} \in \mathbb{N}^\Lambda$ is *output stable* if every $\mathbf{c}' \in \text{post}(\mathbf{c})$ has the same output as \mathbf{c} .

For $\mathbf{o} \in \mathbb{N}^\Gamma$, let $\mathcal{S}_{\mathbf{o}}$ be the set of output stable states with output \mathbf{o} . We say that $\mathbf{c} \in \mathbb{N}^\Lambda$ is *output- \mathbf{o} stabilizing* if $\text{post}(\mathbf{c}) \subseteq \text{pre}(\mathcal{S}_{\mathbf{o}})$. Note that a state is output- \mathbf{o} stabilizing for at most one $\mathbf{o} \in \mathbb{N}^\Gamma$.

Definition 3.15. We say that a CRC \mathcal{C} is *stably computing* if for each input state \mathbf{c} of \mathcal{C} , $\imath(\mathbf{c})$ is output- \mathbf{o} stabilizing for some $\mathbf{o} \in \mathbb{N}^\Gamma$.

If \mathcal{C} is stable deciding, then we say that \mathcal{C} (*stably*) *computes* the function $\varphi : \mathbb{N}^\Sigma \rightarrow \mathbb{N}^\Gamma$ where $\varphi(\mathbf{c}) = \mathbf{o}$ if \mathbf{c} is output- \mathbf{o} stabilizing.

Finally, $\varphi : \mathbb{N}^\Sigma \rightarrow \mathbb{N}^\Gamma$ is called *semilinear* if the set $\{\mathbf{c} \in \mathbb{N}^{\Sigma \cup \Gamma} \mid \varphi(\mathbf{c}|_\Sigma) = \mathbf{c}|_\Gamma\}$ is semilinear.

Example 3.16. Consider again $\min : \mathbb{N}^\Sigma \rightarrow \mathbb{N}^\Gamma$ from Example 3.12. We can easily verify that \min is semilinear. Indeed $\min(x, y) = z$ if and only if $(y = z \wedge x \geq z) \vee (x = z \wedge y \geq z)$. Since $y = z$ is equivalent to $y \leq z \wedge z \leq y$, we observe that $\{(x, y, z) \mid \min(x, y) = z\}$ can be expressed by finite unions and intersections of sets of the form $X_{\mathbf{a}, b} = \{\mathbf{x} \in \mathbb{N}^\Sigma \mid \mathbf{a} \cdot \mathbf{x} \leq b\}$. Indeed, e.g., $y \leq z$ is expressed as $X_{(0, 1, -1), 0}$. Thus \min is semilinear. In the same way we observe that \max is semilinear.

The following is shown in [28] (by using results from [3, 5, 21]).

Theorem 3.17 ([28]). *Stably computing CRCs compute exactly the semilinear functions $\varphi : \mathbb{N}^\Sigma \rightarrow \mathbb{N}^\Gamma$ with $\varphi(\mathbf{0}) = \mathbf{0}$.*

3.4 Speed faults

A reaction α is considered “slow” in a state \mathbf{c} if at least two reactants of α appear in low quantity in \mathbf{c} . We will see in Subsection 4.1 that, assuming the standard stochastic CRN model, the expected time of such reactions to take place is indeed long.

Let α be a uni- or bimolecular reaction. Then α is called *k-fast* for \mathbf{c} , denoted by $\mathbf{c} \Rightarrow_{\alpha, \geq k} \mathbf{c}'$, if $\#_{\mathbf{c}} X \geq k$ for some reactant X of α . Similarly as before, we define $\Rightarrow_{\mathcal{N}, \geq k}$ to denote $\Rightarrow_{\alpha, \geq k}$ for some reaction α of \mathcal{N} and we define the transitive and reflexive closure of $\Rightarrow_{\mathcal{N}, \geq k}$ by $\Rightarrow_{\mathcal{N}, \geq k}^*$. Moreover, we define $\text{pre}_{\mathcal{N}, \geq k}(\mathbf{c}) = \{\mathbf{c}' \in \mathbb{N}^A \mid \mathbf{c}' \Rightarrow_{\mathcal{N}, \geq k}^* \mathbf{c}\}$. As usual we omit the subscript \mathcal{N} , and write simply $\text{pre}_{\geq k}$ when the CRN is clear from the context.

Recall that for stably deciding CRDs it holds that for all input states $\mathbf{c} \in \mathbb{N}^\Sigma \setminus \{\mathbf{0}\}$, we have $\text{post}(\iota(\mathbf{c})) \subseteq \text{pre}(\mathcal{S}_b)$ for some $b \in \{0, 1\}$. In other words, from every state reachable from an input state, we can reach an output- b stable state.

We now define when such a CRD is speed-fault free [20].

Definition 3.18. *We say that a stably deciding CRD with only uni- and bi-molecular reactions is speed-fault free if there is a distinguished input species $F \in \Sigma$ such that for all $k \in \mathbb{N}$ there is an $n \in \mathbb{N}$ such that for all input states $\mathbf{c} \in \mathbb{N}^\Sigma \setminus \{\mathbf{0}\}$ with $\#_{\mathbf{c}} F \geq n$, $\text{post}(\iota(\mathbf{c})) \subseteq \text{pre}_{\geq k}(\mathcal{S}_b)$ for some $b \in \{0, 1\}$.*

The distinguished input species F of Definition 3.18 is called the *fuel* species. Definition 3.18 says that a stably deciding CRD is speed-fault free if any state \mathbf{c}' reachable from some input state having at least n fuel molecules can reach an output-stable state using only k -fast reactions.

The next example is essentially taken from [20].

Example 3.19. *Consider the problem of deciding whether or not there is at least one A_1 or A_2 molecule and no A_3 molecule. Let $\Sigma = \{A_1, A_2, A_3, F\}$, where F is the fuel species. Identify the species A_1, A_2, A_3, F with $X_{100}, X_{010}, X_{001}, X_{000}$, respectively. These subscripts are bit-vectors identifying the presence or absence of the A_i molecules. We introduce species $X_{b_1 b_2 b_3}$ for every bit-vector $b_1 b_2 b_3$ and we introduce bimolecular reactions $X_v + X_w \rightarrow 2X_{\text{OR}(v,w)}$ where $v \neq w$ and OR denotes bitwise OR. For example, we introduce the reactions $X_{000} + X_{001} \rightarrow 2X_{001}$ and $X_{110} + X_{011} \rightarrow 2X_{111}$. One can easily verify that the corresponding CRN \mathcal{N} eventually halts where all molecules are of the same species $X_{b_1 b_2 b_3}$ where the b_i ’s indicate the presence ($b_i = 1$) or absence ($b_i = 0$) of A_i -molecules in the input state. The species $X_{b_1 b_2 b_3}$ with $(b_1 = 1 \vee b_2 = 1) \wedge b_3 = 0$ precisely satisfy the above given predicate and we define these to be exactly the yes-voters. The total number of molecules in a state does not change when reactions take place and so a halting state is nonzero if and only if the input state is nonzero. Hence the obtained CRD \mathcal{D} haltingly (and, therefore, stably) decides the given predicate. Note that the CRD also works fine if we omit the fuel species F and the reactions in which F appears. The sole purpose of species F is to make \mathcal{D} speed-fault free (and, in this way, the ability to “boost”*

the computation by increasing the number of F -molecules). It is easy to see that \mathcal{D} is speed-fault free. Indeed, note that there are 2^3 species $X_{b_1 b_2 b_3}$. Thus at least one species $X_{b_1 b_2 b_3}$ with at least $t/2^3$ molecules, where t be the total number of molecules (recall that t does not change when reactions take place). For any $k \in \mathbb{N}$, take $n := k \cdot 2^3$. Then $t/2^3 \geq n/2^3 = k$, so for any non-halting state \mathbf{c} there is a k -fast reaction for \mathbf{c} , and thus \mathcal{D} is speed-fault free.

The following has been shown in [20].

Theorem 3.20 ([20]). *Let $X \subseteq \mathbb{N}^\Sigma$ for some finite set Σ . Then X is obtained by finite unions, intersections, and complementations of sets of the form $S_A = \{\mathbf{c} \in \mathbb{N}^\Sigma \mid \#_{\mathbf{c}} A = 0\}$ with $A \in \Sigma$ if and only if there is a speed-fault-free stably deciding CRD \mathcal{D} with only uni- and bimolecular reactions that recognizes a set $X' \subseteq \mathbb{N}^{\Sigma'}$ with $X'|_\Sigma = X$ and $\Sigma' = \Sigma \cup \{F\}$ where F is the fuel species of \mathcal{D} .*

So, essentially, such CRDs can only distinguish between existence and non-existence of molecules of the input species. Consequently, they can decide predicates of the form “there is at least one molecule of A ” but *not* predicates of the form “there are at least two molecules of A ”. We note that speed-fault freeness is only an indication (no guarantee) for fast computation. However, the approach described in Example 3.19 to determine (non-)existence of molecules in the input state by speed-fault free CRDs can be shown to fast compute assuming the standard stochastic CRN model (cf. Subsection 4.1), for details see [20].

Remark 3.21. *Non-speed-fault-free stably deciding CRDs \mathcal{D} are not necessarily slower, assuming the standard stochastic CRN model, than speed-fault-free stably deciding CRDs. Indeed, the existence of a single \mathbf{c}' reachable from some input state \mathbf{c} that does not have a k -fast trajectory to an output stable state (i.e., $\mathbf{c}' \in \text{pre}(\mathcal{S}_b)$, but $\mathbf{c}' \notin \text{pre}_{\geq k}(\mathcal{S}_b)$) may have little effect on the speed of the CRD if \mathbf{c}' is unlikely to be reached from the input state \mathbf{c} in the first place.*

The notions and results concerning speed faults turned out to be a stepping stone to prove time lower bounds for problems like leader election (cf. Subsection 4.3) in the standard stochastic CRN model.

Remark 3.22. *In Definition 3.18, the fuel species F is defined to be in the input alphabet Σ . However, in [20] the fuel species is not defined to be in Σ . Consequently, the formulation of Theorem 3.20 above is slightly different than its corresponding formulation in [20]. The formulation of Theorem 3.20 above raises the question whether the whole set X' (not merely $X'|_\Sigma = X$) is obtained by finite unions, intersections, and complementations of sets of the form S_A ’s with $A \in \Sigma'$. Even more generally, one can also consider a definition of speed-fault freeness without explicit fuel species F , where the condition $\#_{\mathbf{c}} F \geq n$ is replaced by $\|\mathbf{c}\| \geq n$.*

Remark 3.23. *We remark that Theorem 3.20 is shown in [20] for the more general class of CRDs which have an initial “context” $\mathbf{c} \in \mathbb{N}^{\Lambda \setminus \Sigma}$ (in fact, this complicates the proof of Theorem 3.20 significantly). This initial context is*

present at the start of the computation along with the input (see Subsection 4.3 for a definition). The notion of a CRD as defined above corresponds to the case where the initial context \mathbf{c} is $\mathbf{0}$.

4 Computing with stochastic chemical reaction networks

The computational CRN models of Section 3 only concern reachability of states, and so their results are independent of stochastics (i.e., how likely a certain state is reached). In this section we recall the well-known standard stochastic model for CRNs and then show, assuming this stochastic model, that CRNs can perform Turing-universal computation if we allow an arbitrary small error probability. We also illustrate the computational mechanism of stochastic CRNs by considering the leader election problem and computing probability distributions.

4.1 Stochastic chemical reaction networks

We first recall the (standard) stochastic model for CRNs [51]. In this section we use several notions and notation from [61].

In the stochastic model for CRNs, each reaction α has a value $k_\alpha \in \mathbb{R}^+$ called the *rate constant*. The *volume* $v \in \mathbb{R}_{>0}$ represents the volume of the well-mixed solution, and as such it determines the expected time for two fixed molecules in the well-mixed solution to meet. The larger the volume, the slower reactions with more than one reactant will take place. Due to physical constraints, the ratio $\|\mathbf{c}\|/v$ is bounded above for well-mixed solutions — this is called the *finite density constraint*. Consequently, one cannot make the volume arbitrarily small, and the cardinality of a state can only grow unboundedly when the volume grows unboundedly too by continuously diluting the solution.

Let $v \in \mathbb{N}$ be a fixed volume. Define the *propensity* $\rho(\mathbf{c}, \alpha)$ of a reaction $\alpha = (\mathbf{r}, \mathbf{p})$ in a state \mathbf{c} as

$$\frac{k_\alpha}{v^{\|\mathbf{r}\|-1}} \prod_{X \in S} \mathbf{c}(X)(\mathbf{c}(X) - 1) \cdots (\mathbf{c}(X) - (\mathbf{r}(X) - 1)),$$

see, e.g., [2]. Here, the product counts the number of ways one can pick all the molecules of \mathbf{r} from the state \mathbf{c} , the fraction $\frac{1}{v^{\|\mathbf{r}\|-1}}$ represents the likelihood that all molecules of \mathbf{r} simultaneously meet, and k_α represents the likelihood that when these molecules meet, they will react (i.e., the reaction will take place).

In particular, if α is unimolecular, then $\rho(\mathbf{c}, \alpha) = k_\alpha \mathbf{c}(X)$ where $X \in \Lambda$ such that $\mathbf{r}(X) = 1$. If α is bimolecular, then $\rho(\mathbf{c}, \alpha)$ is equal to $\frac{k_\alpha}{v} \mathbf{c}(X_1) \mathbf{c}(X_2)$ in the case where $X_1, X_2 \in \Lambda$ are distinct such that $\mathbf{r}(X_1) = \mathbf{r}(X_2) = 1$, and is equal to $\frac{k_\alpha}{v} \mathbf{c}(X)(\mathbf{c}(X) - 1)$ in the case where $\mathbf{r}(X) = 2$.

For $\mathbf{c}, \mathbf{c}' \in \mathbb{N}^A$, define the *transition rate* from \mathbf{c} to \mathbf{c}' as

$$\rho(\mathbf{c}, \mathbf{c}') := \sum_{\alpha \in R(\mathcal{N}), \mathbf{c} \Rightarrow_{\alpha} \mathbf{c}'} \rho(\mathbf{c}, \alpha).$$

We remark that the transition rates define a continuous-time Markov chain on the set of states of \mathcal{N} . However, in this paper we assume no familiarity with Markov chains.

The duration for some reaction to take place within the state \mathbf{c} is an exponential random variable, i.e., a continuous random variable that depends only on the current state and not on the amount of time elapsed (the random variable is “memoryless”), with rate

$$\rho(\mathbf{c}, \mathcal{N}) := \sum_{\alpha \in R(\mathcal{N})} \rho(\mathbf{c}, \alpha).$$

The probability that $\alpha \in R(\mathcal{N})$ is the next reaction to occur in \mathbf{c} is equal to $\rho(\mathbf{c}, \alpha)/\rho(\mathbf{c}, \mathcal{N})$. In particular, the expected time for some reaction to take place in \mathbf{c} is $1/\rho(\mathbf{c}, \mathcal{N})$ (if $\rho(\mathbf{c}, \mathcal{N}) = 0$, then no reaction can occur in \mathbf{c}).

Coming back to the notion of speed faults (cf. Subsection 3.4), we have, in particular, that increasing the molecular count of one of the reactant species of a reaction α , increases its propensity, and therefore decreases the expected time of this reaction to take place (assuming α is the next reaction to take place). In other words, the reaction will indeed be faster.

Example 4.1. Consider a CRN with the reactions $\alpha = A + B \rightarrow C$ and $\beta = 2A \rightarrow C$. For any state \mathbf{c} , we have $\rho(\mathbf{c}, \alpha) = \frac{k_{\alpha}}{v} \mathbf{c}(A) \mathbf{c}(B)$ and $\rho(\mathbf{c}, \beta) = \frac{k_{\beta}}{v} \mathbf{c}(A)(\mathbf{c}(A) - 1)$. Since the expected time for some reaction to take place in \mathbf{c} is $1/\rho(\mathbf{c}, \mathcal{N})$, increasing the number of molecules of A and B will decrease this expected time. If both reactions are applicable to \mathbf{c} (i.e., \mathbf{c} has at least 2 molecules of A and at least 1 molecule of B), then the probability that α is the next reaction to occur in \mathbf{c} is

$$\frac{\rho(\mathbf{c}, \alpha)}{\rho(\mathbf{c}, \alpha) + \rho(\mathbf{c}, \beta)} = \frac{1}{1 + \frac{k_{\beta}}{k_{\alpha}} \cdot \frac{c(A)-1}{c(B)}},$$

which is tending to 1 by increasing the number of B -molecules compared to A -molecules.

Note that because of the finite density constraint, one cannot arbitrarily speed up the computation by decreasing v . Similarly, one cannot arbitrarily increase $\|\mathbf{c}\|$ without increasing v .

Rate constants of chemical reactions are very difficult to control because they depend on the molecular structure of their reactants. Therefore, computational CRN models are often designed to work for any choice of rate constants. That is, we assume we cannot set the rate constants ourselves and so, e.g., the rate constants appear as undetermined constants in various results, such as time

complexity results. For notational convenience, we assume in this paper that all rate constants are equal to some fixed value k . Also, for notational convenience, by a “stochastic CRN” we mean a CRN with rate constants for each reaction that operates in the above described way.

4.2 Turing-universal computation by stochastic chemical reaction networks with possible errors

In this subsection we show that stochastic CRNs can simulate any Turing machine if we allow an arbitrary small nonzero probability of error. Various computational models are Turing universal, and here we follow [61] by simulating deterministic counter automata (which are Turing universal [52, 37]) by stochastic CRNs. See also [61] for a direct simulation of Turing machines by stochastic CRNs.

We briefly recall the notion of a counter automaton (also sometimes called register machine in the literature), see, e.g., [52, 37] for a more elaborate treatment. A (deterministic) *counter automaton* M is a finite state automaton, with distinguished start and halting states q_{start} and q_{halt} , augmented with a finite number of *counters* (also called registers in the literature) which can each hold an arbitrary non-negative integer. For any state q of M , there is an instruction

- $\text{inc}(q, c, q')$ which increments counter c by 1 and then moves to state q' or
- $\text{dec}(q, c, q', q'')$ which either (1) decrements counter c by 1 and moves to state q' if the value of c is nonzero or (2) moves to state q'' (leaving the value of c unchanged) if the value of c is zero.

We assume that for each state q there is exactly one such instruction (hence the adjective “deterministic”). The input of a counter automaton is a nonnegative integer that is stored in the input counter (a distinguished counter) and the input is accepted when, starting in the start state, the computation eventually reaches the halting state.

Given a counter automaton M we define a CRN \mathcal{N}_M simulating M with low probability of error as follows. The set Λ of species of \mathcal{N}_M is equal to $Q \cup C$, where Q is the (finite) set of states and C is the (finite) set of counters of M (we assume without loss of generality that Q and C are disjoint). Furthermore, for each instruction $\text{inc}(q, c, q')$ we introduce the reaction $q \rightarrow c + q'$ in \mathcal{N}_M and for each instruction $\text{dec}(q, c, q', q'')$ we introduce two reactions $q + c \rightarrow q'$ and $q \rightarrow q''$. Let ι be the input counter of M and $\nu \in \mathbb{N}$ be an input value of M . Then we take as the input state \mathbf{i}_ν of \mathcal{N}_M the state with one molecule of the start state q_{start} of M and ν molecules of species ι .

The idea is that during the computation there is exactly one molecule of a species in Q , which represents the current state of M , and, for each $c \in C$, the number of c -molecules is equal to the value of the counter c in M . Note that the reaction $q \rightarrow c + q'$ correctly simulates the instruction $\text{inc}(q, c, q')$. Moreover, if the value of counter c is zero, then $\text{dec}(q, c, q', q'')$ is correctly simulated by $q \rightarrow q''$ (and $q + c \rightarrow q'$ cannot take place). If the value of counter c is nonzero,

then $\text{dec}(q, c, q', q'')$ is correctly simulated by $q + c \rightarrow q'$, however reaction $q \rightarrow q''$ can also take place. In the latter case, i.e., when reaction $q \rightarrow q''$ takes place with c -molecules present, the computation is in error. To make the chance of error arbitrary small, we modify the reaction $q \rightarrow q''$ to make it arbitrary slow: indeed, the slower this reaction, the more likely the correct reaction $q + c \rightarrow q'$ is taken instead. In this way, we trade computation speed for a lower probability of error. To accomplish this trade, the reaction $q \rightarrow q''$ is replaced by the following reactions: $T_i + D \rightarrow T_{i+1} + D$ and $T_{i+1} \rightarrow T_i$ for $i \in \{1, \dots, l-1\}$ and some nonnegative integer l , and $T_1 + q \rightarrow q'' + T_l$. Here, T_1, \dots, T_l , and D are all new species. Moreover, $q + c \rightarrow q'$ is replaced by the reaction $q + c \rightarrow q' + D$. Also, the input state \mathbf{i}_ν of \mathcal{N}_M now also contains one T_l -molecule and a sufficiently large number of D -molecules (depending on the rate constants and volume). The higher the number of D -molecules, the longer it takes for a T_l molecule to convert to a T_1 molecule, while in turn a T_1 molecule is required for the reaction $T_1 + q \rightarrow q'' + T_l$ to take place (which corresponds to moving from state q to state q''). In fact, the production of T_1 takes more and more time as the computation of M progresses since each transition from q to q' by decreasing counter c introduces a new D -molecule. Since the value l is not fixed, we denote the resulting CRN by $\mathcal{N}_{M,l}$.

By [61, Theorem 3.1 and Section 4] we have the following.

Theorem 4.2 ([61]). *Let M be a counter automaton, $\delta > 0$, and $\nu \in \mathbb{N}$. Then there is an $l \in \mathbb{N}$, such that $\mathcal{N}_{M,l}$ on input \mathbf{i}_ν simulates M with a cumulative error probability of at most δ .*

See [61] for upper bounds on the expected computation time and for a faster computation by simulating Turing machines instead of counter automata. Finally, we remark that in [25] an analog of Theorem 4.2 is obtained with error probability zero in terms of “limit-stable” computations: although there might be (infinite) trajectories that lead to an error, these “wrong” trajectories together do not contribute to a positive error probability. While the notions of error probability zero and error-free coincide when each state has only a finite number of reachable states, these notions diverge when states can have an infinite number of reachable states.

We also mention that, independently, a similar approach of simulating Turing machines (via counter automata with multiplication and division) was taken in [4] in the context of population protocols (see Subsection 7.3 for a discussion on the relation between CRNs and population protocols). Finally, we mention that finite circuit computation was shown to be achievable using CRNs in [48] (despite its title, the paper does not show Turing universality).

4.3 Leader election

We now turn to the problem of *leader election*. To motivate this problem, consider a natural extension of the notion of a CRD $\mathcal{D} = (\mathcal{N}, \Sigma, \Lambda_0, \Lambda_1)$, where we extend \mathcal{D} by a vector $j \in \mathbb{N}^{\Lambda \setminus \Sigma}$, called the *context*, to obtain the 5-tuple $\mathcal{D}' = (\mathcal{N}, \Sigma, \Lambda_0, \Lambda_1, j)$. The molecules of j are assumed to be present at the

start of a computation. Hence, the initial state consists of the input molecules and the molecules of the context. It turns out that the notion of a CRD with context does not lead to a (significant) increase in computational power, i.e., CRDs with context can also only compute semilinear sets [5] (the only difference is that CRDs with context can also compute the semilinear sets X with $\mathbf{0} \in X$).

While the expressive power of the class of CRDs with context is equal to that of the class of ordinary CRDs, it is natural to wonder whether or not there are predicates that can compute *faster* using CRDs with context compared to ordinary CRDs.

An interesting special case of this problem is where $\|j\| = 1$, the (unique) molecule of j is called the *leader* of \mathcal{D}' . For designing a CRN that computes a given predicate, it is often convenient to have a leader. Intuitively, a leader can “guide” the computation much like the control flow dictates the computation for an ordinary computer program. Indeed, in [4] it has been shown that various predicates can be computed efficiently if a leader is present. Conversely, various other predicates have been shown to be slow without a leader [10].

In the absence of a leader, one can construct a leader (i.e., a single molecule of some given species) — this is called *leader election*. It is straightforward to elect a leader as follows: assuming there is at least one molecule of $L \in \Sigma$, then the reaction $L + L \rightarrow L$ eventually results in a single L -molecule. However, the process of constructing a leader in this way is slow: $O(n)$ expected time with n molecules of L present in the initial state. Indeed, leader election turns out to be necessarily slow [29].

4.4 Computing probability distributions

A different way to define the computation of a stochastic CRN is through probability distributions [33, 16].

Given a stochastic CRN \mathcal{N} and a state \mathbf{c} , we denote by $\text{Prob}_{\mathcal{N},\mathbf{c}}(t, \mathbf{d})$, for $t \in \mathbb{R}_{\geq 0}$ and state \mathbf{d} , the probability of reaching state \mathbf{d} at time t . Note that for fixed $t_1 \in \mathbb{R}_{\geq 0}$, $\text{Prob}_{\mathcal{N},\mathbf{c}}(t_1, \mathbf{d})$ can be seen as a function sending states $\mathbf{d} \in \mathbb{N}^\Lambda$ to values in the real interval $[0, 1]$. Also note that $\sum_{\mathbf{d} \in \mathbb{N}^\Lambda} \text{Prob}_{\mathcal{N},\mathbf{c}}(t_1, \mathbf{d})$ exists and is equal to 1. We call such functions $f : \mathbb{N}^\Lambda \rightarrow [0, 1]$ with $\sum_{\mathbf{d} \in \mathbb{N}^\Lambda} f(\mathbf{d}) = 1$ *probability mass functions*.

Similarly, if $\pi_{\mathcal{N},\mathbf{c}}(\mathbf{d}) := \lim_{t \rightarrow \infty} \text{Prob}_{\mathcal{N},\mathbf{c}}(t, \mathbf{d})$ exists, then $\pi_{\mathcal{N},\mathbf{c}}(\mathbf{d})$ is a probability mass function. Intuitively, $\pi_{\mathcal{N},\mathbf{c}}(\mathbf{d})$ describes the long-term probability distribution of the states of \mathcal{N} starting from \mathbf{c} .

For probability mass functions f_1 and f_2 , we define $d(f_1, f_2) = \sum_{\mathbf{d} \in \mathbb{N}^\Lambda} |f_1(\mathbf{d}) - f_2(\mathbf{d})|$. Note that $d(f_1, f_2)$ is well defined (in fact, $d(f_1, f_2) \leq 2$). The *support* of a probability mass function f is the set of states \mathbf{c} such that $f(\mathbf{c})$ is nonzero.

The next result shows that arbitrary probability mass functions can be approximated by stochastic CRNs.

Theorem 4.3 ([16]). *Let $f : \mathbb{N}^\Lambda \rightarrow [0, 1]$ be a probability mass function and $\epsilon > 0$. Then there exists a stochastic CRN \mathcal{N} and a state \mathbf{c} such that $\pi_{\mathcal{N},\mathbf{c}}$*

exists and $d(f, \pi_{\mathcal{N}, \mathbf{c}}) < \epsilon$. If f has moreover finite support, then there exists a stochastic CRN \mathcal{N} and a state \mathbf{c} such that $f = \pi_{\mathcal{N}, \mathbf{c}}$.

In the case where $f = \pi_{\mathcal{N}, \mathbf{c}}$, we say that \mathcal{N} *computes* f starting in \mathbf{c} . The proof of Theorem 4.3 first shows the exact computation result (the case where f has finite support) and then observes that the approximate computation result holds since the probability mass functions with finite support are dense for all probability mass functions with countable domain \mathbb{N}^Λ under the distance metric d . It is an open question to characterize the set of probability mass functions that can be (exactly) computed (of course, this set includes all probability mass functions with finite support).

Furthermore, in [16] a *calculus* for probability mass functions that are zero for all but a finite number of states is defined such that any such probability mass function can be obtained from a formula in this calculus. The operators have been shown to be implementable using CRNs [16]. In this way, a *programming language* for probability mass functions based on CRNs is obtained.

5 Computing with continuous chemical reaction networks

5.1 Continuous chemical reaction networks

Until now, a state of a CRN is a vector describing the molecular *counts* $\#X$ of the species X . Such a state is also called a *discrete state*. The larger these molecular counts, the more the stochastic model tends to the continuous mass-action kinetics model, which we call simply the continuous CRN model in this paper, up to some point in time [44]. We remark however that this point in time where divergence of the continuous mass-action kinetics model with the stochastic model can happen is rather soon, namely logarithmic in the number of molecules.

Denote by $\mathbb{R}_{\geq 0}$ the set of nonnegative real numbers. In the continuous CRN model, a state is a $\mathbb{R}_{\geq 0}$ -valued vector describing the molecular *concentrations* $[X]$ of the species X . To distinguish both types of states, we call a state describing molecular concentrations, a *continuous state*. For notational convenience, by a *discrete CRN* (*continuous CRN*, resp.) we mean a CRN that uses discrete (continuous, resp.) states. Note that a stochastic CRN is a particular kind of discrete CRN.

A continuous state evolves continuously (with $\mathbb{R}_{\geq 0}$ -valued time variable t) according to a set of ordinary differential equations (ODEs). To define these ODEs, we first recall the notion of a *stoichiometry matrix* M of \mathcal{N} . The rows and columns of M corresponds to the species X and reactions α of \mathcal{N} , respectively, and each entry $M_{X, \alpha}$ describes the net change of the X -molecules when reaction α takes place. For example, consider the CRN \mathcal{N} with reactions $\alpha = A + B \rightarrow C$

and $\beta = 2C + B \rightarrow 2A + B$. Then the stoichiometry matrix of \mathcal{N} is as follows

$$M = \begin{matrix} & \alpha & \beta \\ \begin{matrix} A \\ B \\ C \end{matrix} & \begin{pmatrix} -1 & 2 \\ -1 & 0 \\ 1 & -2 \end{pmatrix} \end{matrix}.$$

The concentration of some species X changes according to the ODE

$$\frac{d[X]}{dt} = \sum_{\alpha=(\mathbf{r},\mathbf{p}) \in R} k_{\alpha} M_{X,\alpha} \prod_{Y \in \Lambda} [Y]^{\mathbf{r}(Y)},$$

where k_{α} denotes the rate constant of reaction α . Thus each reaction α contributes to a change in concentration of species X that is equal to the product of the reactant concentrations of α , its rate constant k_{α} , and the difference of the number of times X is a product of α minus the number of times X is a reactant of α (so, e.g., the contribution of α to the concentration of X is negative when X appears as a reactant, but not as a product of α).

So, in the given example, a continuous state changes according to the following set of ordinary differential equations:

$$\begin{aligned} \frac{d[A]}{dt} &= -k_{\alpha}[A][B] + 2k_{\beta}[C]^2[B] \\ \frac{d[B]}{dt} &= -k_{\alpha}[A][B] \\ \frac{d[C]}{dt} &= k_{\alpha}[A][B] - 2k_{\beta}[C]^2[B] \end{aligned}$$

where k_{α} and k_{β} are the rate constants of α and β , respectively. We remark that, since states change deterministically in the continuous CRN model, this model is often called the “deterministic” CRN model — however, we do not use this terminology here to avoid possible confusion with the computational CRN models of Section 3 that also have various deterministic aspects.

We now briefly sketch the computational model of continuous CRNs from [30], see that reference for the (involved) formal definition. Roughly speaking, a function $f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ is called *chemically-computable* if there exists a continuous CRN \mathcal{N} and a Λ -indexed vector $q(x)$, where each entry of $q(x)$ is a polynomial in variable x with coefficients from $\mathbb{R}_{\geq 0}$, such that, for all $z \in \mathbb{R}_{\geq 0}$, starting in state $q(z)$, the state \mathbf{c} of the CRN evolves in such a way that $\mathbf{c}(S)$, for some distinguished species S , approaches the value $f(z)$ as $t \rightarrow \infty$. In other words, to compute $f(z)$, q maps z to the initial state of the CRN and the value $f(z)$ is represented by a distinguished entry of the state to which the CRN converges. By using a “dual rail” approach that is similar to the one discussion in Subsection 5.2 below, one can extend the notion of chemically-computable to functions $f : \mathbb{R} \rightarrow \mathbb{R}$, i.e., where the domain and codomain is \mathbb{R} .

It is then shown in [30] that chemically-computable functions are exactly the functions computable by so-called General Purpose Analog Computers as

defined in [11] (which is somewhat different from the original definition in [59]). In turn, General Purpose Analog Computers (as defined in [11]) are computationally equivalent to Turing machines. In this way, this computational model of continuous CRNs is Turing universal.

5.2 Rate-independent computation with continuous chemical reaction networks

Early work on the computational power of continuous CRNs includes [15], where it is shown that various numerical operations such as addition and multiplication can be implemented by continuous CRNs assuming the rate constants of the used reactions can be tuned. Since rate constants are however notoriously difficult to tune, a computational model for continuous CRNs has been introduced in [22] that works independently of the rate constants of the individual reactions. In this subsection we discuss the computational model of [22]. We remark that another rate-independent model of computation for continuous CRNs has been studied in [58].

The computational model for continuous CRNs in [22] is an analog of the computational model in Section 3 but with a reachability function that deals with continuous states instead of (integer-valued) states.

We say that a reaction $\alpha = (\mathbf{r}, \mathbf{p})$ is *applicable* to a continuous state \mathbf{c} if for all species X , $\mathbf{r}(X) > 0$ implies that $\mathbf{c}(X) > 0$. For continuous states \mathbf{c} and \mathbf{d} and $\mathbf{u} \in \mathbb{R}_{\geq 0}^R$, we write $\mathbf{c} \Rightarrow_{\mathbf{u}} \mathbf{d}$ if $\mathbf{c} + M\mathbf{u} = \mathbf{d}$, where M is the stoichiometry matrix, and $\mathbf{u}(\alpha) > 0$ implies that α is applicable to \mathbf{c} . Here $\mathbf{u}(\alpha) \in \mathbb{R}_{\geq 0}$ represents the “amount” of reaction α to occur and so $(M\mathbf{u})(X)$ represents the change in concentration of X when all reactions take place in the amounts described by \mathbf{u} . Therefore, $\mathbf{d} = \mathbf{c} + M\mathbf{u}$ is the state obtained from state \mathbf{c} when the reactions take place according to \mathbf{u} .

We say that \mathbf{d} is *straight-line reachable* from \mathbf{c} , denoted by $\mathbf{c} \Rightarrow \mathbf{d}$, if there is a $\mathbf{u} \in \mathbb{R}_{\geq 0}^R$ such that $\mathbf{c} \Rightarrow_{\mathbf{u}} \mathbf{d}$. As usual, the transitive and reflexive closure of \Rightarrow is denoted by \Rightarrow^* . We say that \mathbf{d} is *segment-reachable* from \mathbf{c} if $\mathbf{c} \Rightarrow^* \mathbf{d}$. Note that segment-reachability is quite different from the reachability notion implied by the ODEs of Subsection 5.1 (which is very much rate dependent). Indeed, Subsection 5.1 implies a definition of reachability such that a continuous state \mathbf{d} is reachable from \mathbf{c} if \mathbf{d} corresponds to the continuous state at time $t > 0$ starting from continuous state \mathbf{c} at time $t = 0$. While the two notions are quite different, [22] shows some relationships between these two notions of reachability. In particular, if a state \mathbf{d} is mass-action reachable from state \mathbf{c} , then \mathbf{d} is segment-reachable from \mathbf{c} .

With the notion of reachability defined in this subsection in place, one can straightforwardly define the continuous analogs of stably deciding for chemical reaction deciders (CRDs) and stably computing for chemical reaction computers (CRCs) of Section 3, see [22]. Let us call the continuous analog of stably computing, $\mathbb{R}_{\geq 0}$ -*stably computing*.

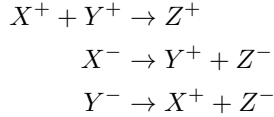
Example 5.1. One verifies that the CRCs described in Examples 3.12 and 3.13 \mathbb{R} -stably compute the $\min(x, y)$ and $\max(x, y)$ functions where x and y are, more generally, in $\mathbb{R}_{\geq 0}$ instead of in \mathbb{N} .

In order to compute general real-valued functions (which allow negative values), we additionally need the notion of a “dual-rail” representation. A *dual-rail representation* of $f : \mathbb{R}^\Sigma \rightarrow \mathbb{R}^\Gamma$ is a function $\hat{f} : \mathbb{R}_{\geq 0}^\Sigma \times \mathbb{R}_{\geq 0}^\Sigma \rightarrow \mathbb{R}_{\geq 0}^\Gamma \times \mathbb{R}_{\geq 0}^\Gamma$ such that for all $\mathbf{x}^+, \mathbf{x}^- \in \mathbb{R}_{\geq 0}^\Sigma$ and $\mathbf{y}^+, \mathbf{y}^- \in \mathbb{R}_{\geq 0}^\Gamma$, $\hat{f}(\mathbf{x}^+, \mathbf{x}^-) = (\mathbf{y}^+, \mathbf{y}^-)$ implies that $f(\mathbf{x}^+ - \mathbf{x}^-) = \mathbf{y}^+ - \mathbf{y}^-$. We remark here that CRNs that compute using dual-rail representations of functions can be straightforwardly composed in contrast to CRNs that compute functions in the ordinary way — this is true for both discrete and continuous CRNs.

The following example is taken from [22].

Example 5.2. Consider the $\min(x, y)$ and $\max(x, y)$ functions over \mathbb{R} , i.e., \min and \max compute the minimum and maximum of two real numbers x and y .

Let $\Sigma = \{X, Y\}$, $\Gamma = \{Z\}$, $\hat{\Sigma} = \{X^+, X^-, Y^+, Y^-\}$, and $\hat{\Gamma} = \{Z^+, Z^-\}$. A dual-rail representation $\widehat{\min} : \mathbb{R}^{\hat{\Sigma}} \rightarrow \mathbb{R}^{\hat{\Gamma}}$ of $\min : \mathbb{R}^\Sigma \rightarrow \mathbb{R}^\Gamma$ can be computed by the reactions



To see this, first notice that both the values $(\#X^+ - \#X^-) + (\#Z^+ - \#Z^-)$ and $(\#Y^+ - \#Y^-) + (\#Z^+ - \#Z^-)$ are invariant under applying these reactions. Also notice that for any state a halting state is reachable: the last two reactions can take place until no X^- and Y^- molecules are present and then the first reaction can take place until the X^+ or Y^+ molecules are exhausted.

Let \mathbf{i} be an initial state, i.e., consisting of only X^+ , X^- , Y^+ , and Y^- molecules. It is easy to see that the CRN has halted in some state \mathbf{c} precisely when $\#_{\mathbf{c}}X^- = \#_{\mathbf{c}}Y^- = 0$ and either $\#_{\mathbf{c}}X^+ = 0$ or $\#_{\mathbf{c}}Y^+ = 0$. By the invariance properties $\#_{\mathbf{i}}X^+ - \#_{\mathbf{i}}X^- = (\#_{\mathbf{i}}X^+ - \#_{\mathbf{i}}X^-) + (\#_{\mathbf{i}}Z^+ - \#_{\mathbf{i}}Z^-) = (\#_{\mathbf{c}}X^+ - \#_{\mathbf{c}}X^-) + (\#_{\mathbf{c}}Z^+ - \#_{\mathbf{c}}Z^-) = \#_{\mathbf{c}}X^+ + (\#_{\mathbf{c}}Z^+ - \#_{\mathbf{c}}Z^-)$ and similarly for $\#_{\mathbf{i}}Y^+ - \#_{\mathbf{i}}Y^-$. In the case where $\#_{\mathbf{c}}X^+ = 0$, we have that $\#_{\mathbf{i}}Y^+ - \#_{\mathbf{i}}Y^- \geq \#_{\mathbf{i}}X^+ - \#_{\mathbf{i}}X^- = \#_{\mathbf{c}}Z^+ - \#_{\mathbf{c}}Z^-$ and so $\#_{\mathbf{c}}Z^+ - \#_{\mathbf{c}}Z^-$ is indeed equal to $\min(\#_{\mathbf{i}}X^+ - \#_{\mathbf{i}}X^-, \#_{\mathbf{i}}Y^+ - \#_{\mathbf{i}}Y^-)$. The case where $\#_{\mathbf{c}}Y^+ = 0$ is analogous, and so we conclude that this CRN (more precisely, CRC) indeed computes $\widehat{\min}$.

Note that the special case where $\#_{\mathbf{i}}X^- = \#_{\mathbf{i}}Y^- = 0$ essentially corresponds to the usual “single-rail” computation of \min , cf. Example 5.1, since then only the reaction $X^+ + Y^+ \rightarrow Z^+$ can take place.

Since $\max(x, y) = -\min(-x, -y)$, a dual-rail representation $\widehat{\max} : \mathbb{R}^{\hat{\Sigma}} \rightarrow \mathbb{R}^{\hat{\Gamma}}$ of $\max : \mathbb{R}^\Sigma \rightarrow \mathbb{R}^\Gamma$ is obtained from $\widehat{\min}$ by reversing the roles of the “plus” and “minus” species (i.e., flipping the superscript). Thus, $\widehat{\max}$ can be computed

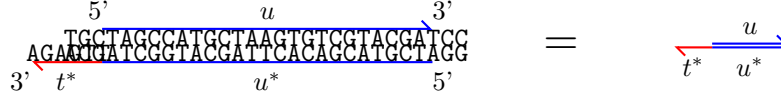


Figure 2: DNA molecule representation.

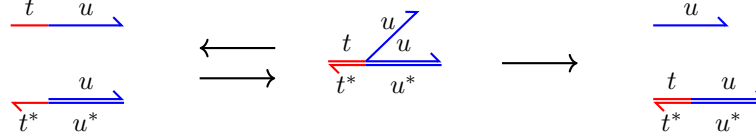
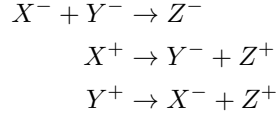


Figure 3: DNA strand displacement.

by the reactions



Let $f : \mathbb{R}^\Sigma \rightarrow \mathbb{R}$ be a function. Then f is called *rational linear* if there is a $\mathbf{a} \in \mathbb{Q}^\Sigma$ such that $f(\mathbf{x}) = \mathbf{a} \cdot \mathbf{x}$, where \cdot denotes the dot product. Moreover, f is called *piecewise rational linear* if there is a finite set S of rational linear functions such that for every $\mathbf{x} \in \mathbb{R}^\Sigma$, $f(\mathbf{x}) = g(\mathbf{x})$ for some $g \in S$.

Theorem 5.3 ([22]). *Let $f : \mathbb{R}^\Sigma \rightarrow \mathbb{R}$ be a function. Then there is a CRC that $\mathbb{R}_{\geq 0}$ -stably computes a dual-rail representation of f if and only if f is continuous and piecewise rational linear.*

It is natural to wonder about the computational complexity of determining whether or not we have $\mathbf{c} \Rightarrow^* \mathbf{d}$ for given continuous states \mathbf{c} and \mathbf{d} . It is shown in [19] that if \mathbf{c} and \mathbf{d} have only rational entries, then this problem can be solved in polynomial time. In contrast, the reachability problem for CRNs using the usual reachability relation for states of Section 2 is much harder, cf. Subsection 7.2.

6 Implementation: DNA strand displacement

In the previous sections we have seen various ways in which (abstract) CRNs can perform computations. We now discuss from [62] a method of implementing an arbitrary (abstract) CRN \mathcal{N} in the wetlab using DNA as a substrate.

First, let us use a concise representation of a DNA molecule that abstracts away from the exact identity of the DNA base-pair sequence. The left-hand

side of Figure 2 depicts a DNA molecule where one single strand consisting of the segments u^* and t^* is bound to the single strand u complementary to u^* (in general, we denote by x^* the Watson-Crick complement of x). As usual, a single strand is denoted by an arrow and its 3'-end is denoted by an arrow head. For visual clarity, we use colors to emphasize the various segments of a single strand/arrow. The concise representation of the left-hand side of Figure 2 is given on the right-hand side of that figure.

We now discuss the key principle of DNA strand displacement, illustrated in Figure 3. Since t and t^* are complementary segments appearing on the left-hand side of Figure 3, these segments can bind, which results in a single DNA molecule given in the middle part of Figure 3. Segment t is a small segment, called a *toehold*, designed to be small enough for the binding to be reversible. Thus, it may happen that t and t^* unbind and we obtain again the situation on the left-hand side of Figure 3. Alternatively, the two u segments may compete for binding with u^* in a random walk fashion and it may happen that the segment u that is connected to t completely pushes out the single strand u that was bound to u^* (single strand u is then called *displaced*), see the right-hand side of Figure 3. Note that this second step of pushing out the single strand u is irreversible.

Figure 4 gives now the implementation of an example reaction $\alpha = A + B \rightarrow C$ using DNA strand displacement from [62]. A molecule of A is represented by a single strand consisting of four segments. The black segment can be arbitrary (although we naturally assume that different segments are always sufficiently different from the (complements of the) other segments of the figure so that they not interfere in unintended ways [27, 47]), and the segments i_A , s_A , o_A together form an identifier for species A . The segments i_A and o_A are toeholds. Aside from these single strands, there are additional molecules L_α and T_α which are assumed to be abundantly present in the well-mixed solution.

If an A -molecule is present, then the “incoming toehold” i_A can bind to its complement i_A^* in molecule L_α . Again, because of the small size of i_A , the single strand representing A may also unbind at this stage. Alternatively, it may compete with the existing single strand $B_\alpha = s_A o_A i_B$ that is part of L_α and possibly push B_α out obtaining H_α . Note that this process is reversible as B_α has i_B as a toehold which can bind to i_B^* in H_α and push out the molecule representing A . Alternatively, if a B -molecule is present, then it may also bind to i_B^* in H_α and this may result in pushing out single strand O_α . The remainder of H_α is waste, and at this stage B_α is waste too. Note that this step is irreversible since O_α cannot bind to the remainder of H_α . Finally, O_α can bind to toehold o_B^* of T_α and this may result in pushing out a single strand that represents C . Again, this step is irreversible. It is important that the first of the three steps is reversible. Indeed, if no B -molecules are present and the first step is irreversible, then A -molecules would be incorrectly consumed by the L_α -molecules.

Since we consume an L_α and a T_α molecule for every application of the reaction α , it is necessary to keep adding these “fuel” molecules to the well-mixed solution to ensure reaction α can keep taking place.

Additional systematic methodologies for compiling a given (abstract) CRN

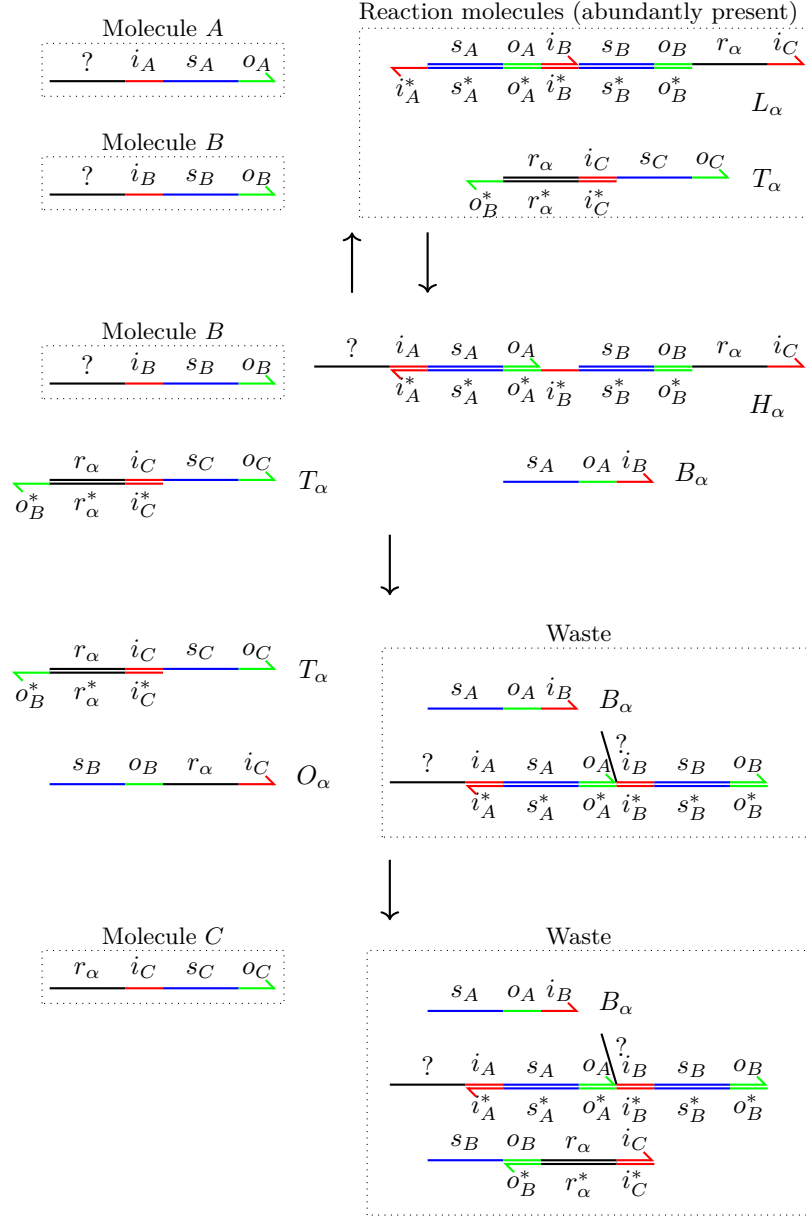


Figure 4: Simulating reaction $\alpha = A + B \rightarrow C$ through DNA strand displacements.

to an implementation are given in [23, 7]. Note that an implementation of an (abstract CRN) is a CRN as well, called an *implementation CRN*. Verification of correctness of an implementation CRN against an abstract CRN has been studied using the notion of pathway decomposition in [60] and using the notion of bisimulation in [42]. We also remark that the notion of correctness in general depends on the computational model that is assumed. For example, the implementation CRN of Figure 4 would in general not faithfully represent the original abstract CRN if we assume a computational model that highly depends on specific values of the rate constants, like the model of computing with probability density functions in Subsection 4.4.

Note that for CRNs \mathcal{N} having “non mass-conserving” reactions like $\mathbf{0} \rightarrow A$ or $A \rightarrow \mathbf{0}$, where $\mathbf{0}$ is a zero vector, we necessarily need “fuel” molecules or “waste” molecules, respectively, for any implementation of \mathcal{N} in nature (notice that the above mentioned implementation of a CRN by DNA strand displacement needs fuel molecules and has waste molecules also for mass-conserving reactions). Mass-conserving CRNs are exactly the CRNs for which it is possible to assign positive integers to the species, a weight vector \mathbf{v} , such that the weighted sum of the molecules in a state is invariant under the application of any reaction (i.e., $\mathbf{v}^T M = \mathbf{0}^T$, where M is the stoichiometry matrix of the CRN). Such a weight vector is called a *conservation vector* [39].

7 Related research fields

Since CRNs form a mathematically natural model, it is not surprising that this notion (or notions very similar to it) has also appeared in other contexts. Indeed, CRNs are very closely related to the notions of Petri nets [53, 56] and vector addition systems [43] from the theory of concurrency and population protocols [6] from the theory of distributed computing.

7.1 Petri nets

We first turn to Petri nets [53, 56], which are nearly identical to CRNs. In a Petri net, molecules are called *tokens*, species are called *places*, reactions are called *transitions*, and states are called *markings*. The *firing* of a transition in a Petri net corresponds to a reaction that takes place in a CRN. More advanced notions often also have their counterpart in Petri net theory (and vice versa), e.g., the notion of a conservation vector (mentioned in Section 6) is called a *P-invariant* in Petri net theory.

A Petri net is defined as a directed bipartite multigraph where the vertices of one colour class P are the places (and are depicted as round vertices) and the vertices of the other colour class T are the transitions (and are depicted as square vertices).

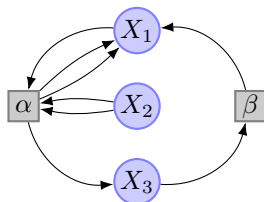
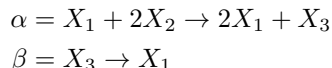


Figure 5: The Petri net corresponding to the CRN \mathcal{N} of Example 7.1.

Example 7.1. Consider the CRN $\mathcal{N} = (\{X_1, X_2, X_3\}, \{\alpha, \beta\})$ with



The Petri net corresponding to \mathcal{N} is depicted in Figure 5. Note that the reactions correspond to square vertices (i.e., transitions) and that the species correspond to round vertices (i.e., places) in Figure 5. The reactants (products, resp.) of each reaction correspond to the incoming (outgoing, resp.) arrows, with multiplicity, of the corresponding transition. For example since, $X_1 + 2X_2$ are the reactants of α , there is one arrow from X_1 to α and two arrows from X_2 to α .

There are some small differences between the (usual) definitions of a CRN and a Petri net, which for most problems are irrelevant. One difference is that a Petri net has an initial marking (i.e., an initial state), while this is not the case in the (usual) definition of a CRN. Of course, such a fixed initial state can be useful in the context of CRNs as well (see, e.g., Subsection 4.4). A more subtle difference is that a Petri net may have two (or more) transitions with the same multisets of incoming and outgoing arrows, which would corresponds to two distinct reactions of the form (\mathbf{r}, \mathbf{p}) .

Many results on Petri nets deal with behavioral properties starting from the initial marking/state. Roughly speaking, a Petri net is most often considered as a *generator* of states. In contrast, the computational CRN models of Section 3 deal with accepting or rejecting an unknown input state. The models from Section 3 have not been considered in the context of Petri net theory but have been taken from the theory of population protocols, cf. Subsection 7.3 below.

The notion of a *stochastic Petri net* [8, 9, 49] studied in the literature is similar to the notion of a stochastic CRN, however the notion of propensity (defined in Subsection 4.1) that is used is different. More specifically, in stochastic Petri nets the propensity is *equal* to the rate constant (called *firing rate* in the context of stochastic Petri nets) and so, e.g., transition t can fire for markings \mathbf{c} and \mathbf{d} , then the expected time for t to fire for \mathbf{c} is equal to the expected time for t to fire for \mathbf{d} — this behavior is, of course, very different from stochastic CRNs (indeed, for stochastic CRNs the expected time for a reaction to take place decreases when increasing the molecule counts of species that appear as reactants of that reaction).

The notion of a *continuous Petri net* [55, 26] studied in the literature is quite different from the notion of a continuous CRN from Subsection 5.1. Indeed, the former is not based on differential equations, but instead allows transitions/reactions to be applied “ $x \in \mathbb{R}$ times”. In this way, continuous Petri nets are more related to the rate-independent mode of operation discussed in Subsection 5.2.

We mention that there are other classes of Petri nets, like hybrid Petri nets [26] and coloured Petri nets [40], which currently have not yet been considered in the context of CRNs.

7.2 Vector addition systems

A *vector addition system* (VAS for short) [43] is a finite subset A of \mathbb{Z}^Λ , where \mathbb{Z} is the set of integers and Λ is finite. The elements of A are called *actions*. Similar as for CRNs and Petri nets, a *state* is an element of \mathbb{N}^Λ . An action $\mathbf{a} \in A$ can *fire* for state \mathbf{c} if $\mathbf{c} + \mathbf{a}$ is a state.

For an action \mathbf{a} , let $\mathbf{p}, \mathbf{r} \in \mathbb{N}^\Lambda$ be such that for all $X \in \Lambda$, we have (1) $\mathbf{p}(X) = \mathbf{a}(X)$ and $\mathbf{r}(X) = 0$ if $\mathbf{a}(X) \geq 0$ and (2) $\mathbf{r}(X) = -\mathbf{a}(X)$ and $\mathbf{p}(X) = 0$ otherwise. Then $\mathbf{a} = \mathbf{p} - \mathbf{r}$, and we can easily see that the reaction (\mathbf{r}, \mathbf{p}) simulates the action \mathbf{a} . In this way, for each VAS A there is a CRN that simulates A .

There is, however, an issue in simulating a CRN by a VAS. Consider the VAS A obtained from a CRN \mathcal{N} by replacing each reaction (\mathbf{r}, \mathbf{p}) by the action $\mathbf{p} - \mathbf{r}$. Then A may behave *differently* than \mathcal{N} . Indeed, for example, the reaction $A + 3B \rightarrow C + 3B$ cannot be applied to a state \mathbf{d} with only the single molecule A , but the action $\mathbf{p} - \mathbf{r} = C - A$ can be applied to \mathbf{d} . More generally, the construction does not work for “catalyst-like” reactions (\mathbf{r}, \mathbf{p}) , where $\mathbf{r}(X)$ and $\mathbf{p}(X)$ are both nonzero for some species X . However, for each CRN \mathcal{N} there is a CRN \mathcal{N}' without catalyst-like reactions that behaves very similar to \mathcal{N} . The CRN \mathcal{N}' is obtained from \mathcal{N} by introducing a new species Q_α for each catalyst-like reaction $\alpha = (\mathbf{r}, \mathbf{p})$ and replacing α by the reactions $\alpha_1 = (\mathbf{r}, \mathbf{q}_\alpha)$ and $\alpha_2 = (\mathbf{q}_\alpha, \mathbf{p})$, where \mathbf{q}_α contains one copy of Q_α and nothing else [61, 24]. In this way, e.g., the reaction $\alpha = A + 3B \rightarrow C + 3B$ is simulated by the reactions $A + 3B \rightarrow Q_\alpha$ and $Q_\alpha \rightarrow C + 3B$. For many problems the difference between \mathcal{N} and \mathcal{N}' is irrelevant and for these problems we can equivalently consider the VAS corresponding to \mathcal{N}' .

The reachability problem for VASs, i.e., to determine for given states \mathbf{c} and \mathbf{d} whether or not \mathbf{d} can be reached from \mathbf{c} , has been intensively investigated and is well known to be EXPSPACE-hard [18] (lower bound) and decidable [50, 45] (upper bound, see the introduction of [45] for a more detailed historical account of the decidability proofs). By the above, these results directly carry over to the domains of Petri nets and CRNs.

7.3 Population protocols

The notion of a *population protocol* was introduced in [3] as a model for distributed computing. A population protocol models a finite set of *agents* that each hold a *state* from a fixed finite set Q of states. When two agents bump into each other, the agents change their state according to a transition function $\delta : Q^2 \rightarrow Q^2$. Agents with a common state are indistinguishable, so a particular global state of a set of agents can be described as a multiset of the states of the agents. We can now easily see that population protocols correspond to CRNs where each reaction (\mathbf{r}, \mathbf{p}) is such that $\|\mathbf{r}\| = \|\mathbf{p}\| = 2$. Indeed, agents correspond to molecules, states correspond to species, and if $\delta(q_1, q_2) = (q_3, q_4)$, then this corresponds to reaction $q_1 + q_2 \rightarrow q_3 + q_4$. More precisely, the class of population protocols therefore actually corresponds to the class of CRNs where each reaction (\mathbf{r}, \mathbf{p}) is such that $\|\mathbf{r}\| = \|\mathbf{p}\| = 2$ *and*, additionally, there is a reaction for each pair of species. However, we may have $\delta(q_1, q_2) = (q_1, q_2)$ and so the corresponding reaction (\mathbf{r}, \mathbf{p}) is mute. For most problems the existence or absence of mute reactions is irrelevant. The computational CRN model of Subsection 3.2 is the natural generalization to CRNs of the original computational model for population protocols from [3]. The interpretation of the computational model of Subsection 3.2 in terms of population protocols is as follows: each agent starts with an input state and “eventually” there is agreement among the agents of accepting the input or not (we use eventually in the sloppy way here, cf. Remark 3.3: we actually mean “during the computation it is always possible to reach a state where”). Some states are designated as “yes” states and others as “no” states — in this way, agents communicate their opinion. Notice that the notion of stably deciding is natural within the context of population protocols since agents will keep bumping into each other, triggering the application of the transition function δ . Indeed, stably deciding (not haltingly deciding) is the most studied mode of operation for population protocols.

Because the class of population protocols corresponds to a proper subclass of all CRNs, results concerning population protocols do not necessarily hold for the whole class of CRNs. One particularly important aspect of population protocols is that the number of agents stay fixed during a computation. In other words, in the corresponding CRN \mathcal{N} , we have that for all states \mathbf{c} and $\mathbf{d} \in \text{post}(\mathbf{c})$, $\|\mathbf{d}\| = \|\mathbf{c}\|$. Since there are only a finite number of states of a given size, we have that the set $\text{post}(\mathbf{c})$ is finite. It is easy to define CRNs that violate this property: take, e.g., a CRN having the reaction $\mathbf{0} \rightarrow A$, where $\mathbf{0}$ is a zero vector.

The efficiency of population protocol algorithms is expressed in terms of the expected number of interactions between agents, where the two agents for each interaction are chosen at random. While the model is therefore similar to that of stochastic CRNs where the rate constants are all 1 and the volume is equal to the number of agents, there is a difference in that population protocols use discrete time and stochastic CRNs use continuous time.

8 Discussion

The goal of this paper is to introduce in a tutorial fashion the basic concepts and results concerning computational CRNs as well as to review some of the main strands of research in this area. By now the literature of this research field is really vast, so it is not possible to cover (in a space-limited tutorial) all interesting research directions. We complete this tutorial by mentioning a few research directions that we did not cover.

Most prominently, we have not discussed the important topic of model checking, i.e., verifying behavioral properties of CRNs. CRN theory, see, e.g., [31, 32, 38, 35], is a well-established research field that is traditionally used to study CRNs occurring in nature, but it can equally well be used to model check human-designed computational CRNs. Model checking techniques can be drawn from various contexts. Indeed, for example, various notions introduced originally in the context of continuous CRNs, such as the important notion of deficiency, have found their use also for discrete CRNs [1]. As another example, notions introduced originally in the context of Petri nets, such as the notion of a T-invariant, have found their use for discrete CRNs, see, e.g., [13].

Also, since CRNs behave inherently asynchronously, it is natural to link CRNs to asynchronous logic circuits. This research direction is pursued in [17] where (among other results) it is shown that the Muller C-element (a fundamental asynchronous component) [63] can be simulated by a CRN. Other work on asynchronous logic circuits implemented by CRNs includes [57].

CRNs have a finite number of reactions and a finite number of species. It would be interesting to see what results concerning CRNs hold in the more general setting where we drop one or both of these assumptions (of course, allowing an infinite number of species without allowing an infinite number of reactions makes little sense). Motivated by polymers, which can be of arbitrary length, a special class of CRNs with an infinite number of species has been considered in [41].

In this paper we have also assumed that CRNs reside in well-mixed solutions. However, one can also consider non-homogeneous environments. For example, in [54] it is shown possible to implement CRNs tied to surfaces, which are called *surface CRNs*. The “spatial awareness” of surface CRNs results in a higher computational expressivity compared to (the usual) CRNs that reside in well-mixed solutions. Indeed, surface CRNs can simulate arbitrary Turing machines (without any theoretical probability of error) [54].

Acknowledgments

We thank Dave Doty, Grzegorz Rozenberg, David Soloveichik, and three anonymous referees for many useful comments on earlier versions of this paper. R.B. is a postdoctoral fellow of the Research Foundation – Flanders (FWO).

References

- [1] D. F. Anderson, G. A. Enciso, and M. D. Johnston. Stochastic analysis of biochemical reaction networks with absolute concentration robustness. *Journal of The Royal Society Interface*, 11(93), 2014.
- [2] D. F. Anderson and T. G. Kurtz. Continuous time Markov chain models for chemical reaction networks. In H. Koepl, G. Setti, M. di Bernardo, and D. Densmore, editors, *Design and Analysis of Biomolecular Circuits: Engineering Approaches to Systems and Synthetic Biology*, pages 3–42. Springer New York, 2011.
- [3] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006.
- [4] D. Angluin, J. Aspnes, and D. Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, 2008.
- [5] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.
- [6] J. Aspnes and E. Ruppert. An introduction to population protocols. *Bulletin of the EATCS*, 93:98–117, 2007.
- [7] S. Badelt, S. W. Shin, R. F. Johnson, Q. Dong, C. Thachuk, and E. Winfree. A general-purpose CRN-to-DSD compiler with formal verification, optimization, and simulation capabilities. In R. Brijder and L. Qian, editors, *Proceedings of the 23th International Conference on DNA Computing and Molecular Programming (DNA 23)*, volume 10467 of *Lecture Notes in Computer Science*, pages 232–248. Springer, 2017.
- [8] G. Balbo. Introduction to stochastic Petri nets. In E. Brinksma, H. Hermanns, and J. Katoen, editors, *Lectures on Formal Methods and Performance Analysis*, volume 2090 of *Lecture Notes in Computer Science*, pages 84–155. Springer, 2000.
- [9] F. Bause and P. S. Kritzinger. *Stochastic Petri nets: An Introduction to the Theory, Second Edition*. Vieweg Verlag, 2002.
- [10] A. Belleville, D. Doty, and D. Soloveichik. Hardness of computing and approximating predicates and functions with leaderless population protocols. In I. Chatzigiannakis, P. Indyk, F. Kuhn, and A. Muscholl, editors, *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *LIPIcs*, pages 141:1–141:14. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2017.
- [11] O. Bournez, M. L. Campagnolo, D. S. Graça, and E. Hainry. Polynomial differential equations compute all real computable functions on computable compact intervals. *Journal of Complexity*, 23(3):317 – 335, 2007.

- [12] R. Brijder. Minimal output unstable configurations in chemical reaction networks and deciders. *Natural Computing*, 15(2):235–244, 2016.
- [13] R. Brijder. Dominance and deficiency for Petri nets and chemical reaction networks. *Natural Computing*, 16(2):285–294, 2017.
- [14] R. Brijder, D. Doty, and D. Soloveichik. Democratic, existential, and consensus-based output conventions in stable computation by chemical reaction networks. *Natural Computing*, 17(1), 2018.
- [15] H. J. Buisman, H. M. M. ten Eikelder, P. A. J. Hilbers, and A. M. L. Liekens. Computing algebraic functions with biochemical reaction networks. *Artificial Life*, 15(1):5–19, 2009.
- [16] L. Cardelli, M. Kwiatkowska, and L. Laurenti. Programming discrete distributions with chemical reaction networks. In Y. Rondelez and D. Woods, editors, *Proceedings of the 22th International Conference on DNA Computing and Molecular Programming (DNA 22)*, volume 9818 of *Lecture Notes in Computer Science*, pages 35–51. Springer, 2016.
- [17] L. Cardelli, M. Kwiatkowska, and M. Whitby. Chemical reaction network designs for asynchronous logic circuits. In Y. Rondelez and D. Woods, editors, *Proceedings of the 22th International Conference on DNA Computing and Molecular Programming (DNA 22)*, volume 9818 of *Lecture Notes in Computer Science*, pages 67–81. Springer, 2016.
- [18] E. Cardoza, R. J. Lipton, and A. R. Meyer. Exponential space complete problems for Petri nets and commutative semigroups: Preliminary report. In A. K. Chandra, D. Wotschke, E. P. Friedman, and M. A. Harrison, editors, *Proceedings of the 8th Annual ACM Symposium on Theory of Computing (STOC 1976)*, pages 50–54. ACM, 1976.
- [19] A. Case, J. H. Lutz, and D. M. Stull. Reachability problems for continuous chemical reaction networks. In M. Amos and A. Condon, editors, *Proceedings of the 15th International Conference on Unconventional Computation and Natural Computation (UCNC 2016)*, volume 9726 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2016.
- [20] H.-L. Chen, R. Cummings, D. Doty, and D. Soloveichik. Speed faults in computation by chemical reaction networks. *Distributed Computing*, 30(5):373–390, 2017.
- [21] H.-L. Chen, D. Doty, and D. Soloveichik. Deterministic function computation with chemical reaction networks. *Natural Computing*, 13(4):517–534, 2014.
- [22] H.-L. Chen, D. Doty, and D. Soloveichik. Rate-independent computation in continuous chemical reaction networks. In M. Naor, editor, *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science (ITCS’14)*, pages 313–326. ACM, 2014.

- [23] Y.-J. Chen, N. Dalchau, N. Srinivas, A. Phillips, L. Cardelli, D. Soloveichik, and G. Seelig. Programmable chemical controllers made from DNA. *Nature Nanotechnology*, 8:755–762, 2013.
- [24] M. Cook, D. Soloveichik, E. Winfree, and J. Bruck. Programmability of chemical reaction networks. In A. Condon, D. Harel, J. N. Kok, A. Salomaa, and E. Winfree, editors, *Algorithmic Bioprocesses*, pages 543–584. Springer Berlin Heidelberg, 2009.
- [25] R. Cummings, D. Doty, and D. Soloveichik. Probability 1 computation with chemical reaction networks. *Natural Computing*, 15(2):245–261, 2016.
- [26] R. David and H. Alla. *Discrete, Continuous, and Hybrid Petri Nets*. Springer, second edition edition, 2010.
- [27] R. Dirks, J. Bois, J. Schaeffer, E. Winfree, and N. Pierce. Thermodynamic analysis of interacting nucleic acid strands. *SIAM Review*, 49(1):65–88, 2007.
- [28] D. Doty and M. Hajiaghayi. Leaderless deterministic chemical reaction networks. *Natural Computing*, 14(2):213–223, 2015.
- [29] D. Doty and D. Soloveichik. Stable leader election in population protocols requires linear time. *Distributed Computing*, 31:257–271, 2018.
- [30] F. Fages, G. L. Guludec, O. Bournez, and A. Pouly. Strong Turing completeness of continuous chemical reaction networks and compilation of mixed analog-digital programs. In J. Feret and H. Koepl, editors, *15th International Conference on Computational Methods in Systems Biology (CMSB 2017)*, volume 10545 of *Lecture Notes in Computer Science*, pages 108–127. Springer, 2017.
- [31] M. Feinberg. Lectures on chemical reaction networks, 1980. URL: <https://crnt.osu.edu/LecturesOnReactionNetworks>.
- [32] M. Feinberg and F. Horn. Chemical mechanism structure and the coincidence of the stoichiometric and kinetic subspaces. *Archive for Rational Mechanics and Analysis*, 66(1):83–97, 1977.
- [33] B. Fett, J. Bruck, and M. D. Riedel. Synthesizing stochasticity in biochemical systems. In *Proceedings of the 44th Annual Design Automation Conference (DAC 2007)*, pages 640–645. ACM, 2007.
- [34] S. Ginsburg and E. H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific Journal of Mathematics*, 16(2):285–296, 1966.
- [35] J. Gunawardena. Chemical reaction network theory for in-silico biologists, 2003. URL: <http://vcp.med.harvard.edu/papers/crnt.pdf>.

- [36] J. E. Hopcroft and J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8:135–159, 1979.
- [37] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
- [38] F. Horn. Necessary and sufficient conditions for complex balancing in chemical kinetics. *Archive for Rational Mechanics and Analysis*, 49(3):172–186, 1972.
- [39] F. Horn and R. Jackson. General mass action kinetics. *Archive for Rational Mechanics and Analysis*, 47(2):81–116, 1972.
- [40] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. Springer-Verlag, second edition edition, 1996.
- [41] R. Johnson and E. Winfree. Verifying polymer reaction networks using bisimulation, 2014. URL: <http://www.dna.caltech.edu/Papers/Polymers2014-VEMDP.pdf>.
- [42] R. F. Johnson, Q. Dong, and E. Winfree. Verifying chemical reaction network implementations: A bisimulation approach. In Y. Rondelez and D. Woods, editors, *Proceedings of the 22th International Conference on DNA Computing and Molecular Programming (DNA 22)*, volume 9818 of *Lecture Notes in Computer Science*, pages 114–134. Springer, 2016.
- [43] R. M. Karp and R. E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, 1969.
- [44] T. G. Kurtz. The relationship between stochastic and deterministic models for chemical reactions. *The Journal of Chemical Physics*, 57(7):2976–2978, 1972.
- [45] J. Leroux. Vector addition systems reachability problem (a simpler solution). In A. Voronkov, editor, *Proceedings of the Alan Turing Centenary Conference (Turing-100)*, volume 10 of *EPiC Series*, pages 214–228, 2012.
- [46] A. M. L. Liekens and C. T. Fernando. Turing complete catalytic particle computers. In F. Almeida e Costa, L. M. Rocha, E. Costa, I. Harvey, and A. Coutinho, editors, *Proceedings of the 9th European Conference on Artificial Life (ECAL 2007)*, volume 4648 of *Lecture Notes in Computer Science*, pages 1202–1211. Springer, 2007.
- [47] R. Lorenz, S. H. Bernhart, C. Höner zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker. ViennaRNA package 2.0. *Algorithms for Molecular Biology*, 6(1):26, 2011.
- [48] M. O. Magnasco. Chemical kinetics is Turing universal. *Physical Review Letters*, 78:1190–1193, 1997.

- [49] M. A. Marsan. Stochastic Petri nets: an elementary introduction. In G. Rozenberg, editor, *Advances in Petri Nets 1989*, volume 424 of *Lecture Notes in Computer Science*, pages 1–29. Springer, 1988.
- [50] E. W. Mayr. An algorithm for the general Petri net reachability problem. *SIAM Journal on Computing*, 13(3):441–460, 1984.
- [51] D. A. McQuarrie. Stochastic approach to chemical kinetics. *Journal of Applied Probability*, 4(3):413–478, 1967.
- [52] M. L. Minsky. Recursive unsolvability of Post’s problem of “tag” and other topics in theory of Turing machines. *Annals of Mathematics*, 74(3):437–455, 1961.
- [53] J. L. Peterson. Petri nets. *ACM Computing Surveys*, 9(3):223–252, 1977.
- [54] L. Qian and E. Winfree. Parallel and scalable computation and spatial dynamics with DNA-based chemical reaction networks on a surface. In S. Murata and S. Kobayashi, editors, *Proceedings of the 20th International Conference on DNA Computing and Molecular Programming (DNA 20)*, volume 8727 of *Lecture Notes in Computer Science*, pages 114–131. Springer, 2014.
- [55] L. Recalde, E. Teruel, and M. S. Suárez. Autonomous continuous P/T systems. In S. Donatelli and H. C. M. Kleijn, editors, *Proceedings of the 20th International Conference on the Applications and Theory of Petri Nets (ICATPN ’99)*, volume 1639 of *Lecture Notes in Computer Science*, pages 107–126. Springer, 1999.
- [56] W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer, 1998.
- [57] S. A. Salehi, M. D. Riedel, and K. K. Parhi. Asynchronous discrete-time signal processing with molecular reactions. In *Proceedings of the 48th Asilomar Conference on Signals, Systems and Computers*, pages 1767–1772, 2014.
- [58] P. Senum and M. Riedel. Rate-independent constructs for chemical computation. *PLOS ONE*, 6(6):1–12, 06 2011.
- [59] C. E. Shannon. Mathematical theory of the differential analyzer. *Journal of Mathematics and Physics*, 20(1-4):337–354, 1941.
- [60] S. W. Shin, C. Thachuk, and E. Winfree. Verifying chemical reaction network implementations: A pathway decomposition approach. *Theoretical Computer Science*, 2017. Online first.
- [61] D. Soloveichik, M. Cook, E. Winfree, and J. Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing*, 7(4):615–633, 2008.

- [62] D. Soloveichik, G. Seelig, and E. Winfree. DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, 107(12):5393–5398, 2010.
- [63] J. Sparso and S. Furber, editors. *Principles of Asynchronous Circuit Design: A Systems Perspective*. Springer, 2001.