Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Verifying polymer reaction networks using bisimulation

Robert F. Johnson [a,*], Erik Winfree [b]

[a] Bioengineering, California Institute of Technology, United States of America
[b] Computer Science & Bioengineering, California Institute of Technology, United States of America

ARTICLE INFO

Article history:
Received 8 January 2020
Received in revised form 6 August 2020
Accepted 10 August 2020
Available online xxxx
Communicated by L. Cardelli

Keywords:
Verification
Molecular computing
Bisimulation
Chemical Reaction Networks
Polymer Reaction Networks
DNA strand displacement

ABSTRACT

The Chemical Reaction Network model has been proposed as a programming language for molecular programming. Methods to implement arbitrary CRNs using DNA strand displacement circuits have been investigated, as have methods to prove the correctness of those or other implementations. However, the stochastic Chemical Reaction Network model is provably not deterministically Turing-universal, that is, it is impossible to create a stochastic CRN where a given output molecule is produced if and only if an arbitrary Turing machine accepts. A DNA stack machine that can simulate arbitrary Turing machines with minimal slowdown deterministically has been proposed, but it uses unbounded polymers that cannot be modeled as a Chemical Reaction Network. We propose an extended version of a Chemical Reaction Network that models unbounded linear polymers made from a finite number of monomers. This Polymer Reaction Network model covers the DNA stack machine, as well as copy-tolerant Turing machines and some examples from biochemistry. We adapt the bisimulation method of verifying DNA implementations of Chemical Reaction Networks to our model, and use it to prove the correctness of the DNA stack machine implementation. We define a subclass of single-locus Polymer Reaction Networks and show that any member of that class can be bisimulated by a network using only four primitives, suggesting a method of DNA implementation. Finally, we prove that deciding whether an implementation is a bisimulation is Π^0_2-complete, and thus undecidable in the general case, although it is tractable in many special cases of interest. We hope that the ability to model and verify implementations of Polymer Reaction Networks will aid in the rational design of molecular systems.

1. Introduction

We consider the problem of how molecules can compute: how do biological systems use their components to compute, and what computing systems can be built with biological or bio-compatible molecules? For relatively small molecules in a well-mixed solution, the well-studied Chemical Reaction Network (CRN) model is a natural way to describe them. Known examples of computation with CRNs include useful small devices such as the approximate majority CRN [3,17] and the rock-paper-scissors oscillator [32,20,49], boolean circuits [38] and neural networks [25], as well as more general results, including deterministic computation of arbitrary semilinear functions [2,15,21] and simulation of Turing machines with arbitrarily small error probability [47]. (For those not interested in computation per se, Turing universality may be taken as an assurance that these systems are capable of a wide class of complex behaviors.) Further, computationally interesting (or

* Corresponding author.
E-mail address: rfjohnso@dna.caltech.edu (R.F. Johnson).

https://doi.org/10.1016/j.tcs.2020.08.007

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

uninteresting) CRNs can be "compiled" into physical devices: Soloveichik et al. [48], Qian et al. [42], and Cardelli [9], among others, give schemes to construct a DNA Strand Displacement (DSD) circuit that implements an arbitrary CRN.

One assumption of the CRN model is that the molecules are in a "well-mixed solution": that there is no concept of geometry or spatial organization of the molecules, that any pair of molecules is as likely to interact as any other, and that the only relevant information about the current state is the count (or concentration) of each molecule present. For small circuits like the ones mentioned above, this is quite reasonable. For classic models of computation and for biological systems, however, this assumption doesn't match: Turing machines, DNA/RNA/Proteins, and the cytoskeleton are all fundamental examples and fundamentally geometric. A counting argument suggests why: consider a system with k "types of object" (e.g. chemical species, Turing machine tape symbols) and a state of that system with n total objects. In a well-mixed CRN, the number of possible such states is on the order of (but less than) n^k; in a Turing machine or other geometric system, that number is on the order of k^n. In uniform computation—a single machine built to handle arbitrarily large computations—we have a constant k with n scaling with the size of the computation; so for example, the CRN that simulates Turing machines mentioned above uses around 3^n copies of a given molecule to simulate a Turing machine with n tape squares filled [47].

For such reasons, researchers have begun building molecular computing systems that take advantage of geometry. There are a number of variations on the concept of a DNA walker moving around a surface, often DNA origami, in a programmable way; a particularly complex example is the cargo-sorting robot of Thubagere et al. [52]. Chatterjee et al. have built logic circuits on origami, using a constant number of components regardless of the size of the circuit [14]. In the examples closest to abstract CRNs, Cardelli and Zavattaro discussed a CRN-like model with association and dissociation [13]; Qian et al. proposed a DNA implementation of a generic stack machine [42]; and Qian and Winfree proposed a DNA implementation of CRN-like reactions localized on a surface [43].

Also relevant to this topic are theoretical results on the computational power of well-mixed CRNs, and the difference in power between well-mixed CRNs and geometry-using models. The two most relevant results are that well-mixed CRNs that "always eventually" compute the right answer (in a certain sense well-defined in the theorem) can compute exactly the semilinear functions [2,15], and that the reachability problem for CRNs is decidable by a Turing machine [29,36]. The reachability problem is in an informal sense the CRN equivalent of the Turing machine halting problem; but since the Turing machine halting problem is undecidable, any CRN trying to simulate a Turing machine must have some reachable state that involves an error. Thus those CRNs that try to simulate Turing machines can either do so deterministically and space-efficiently in a non-uniform sense, where a single CRN can simulate a Turing machine with a given bound on its tape size by using separate species for each tape position, and thus a larger CRN must be created to simulate a larger Turing machine tape [38,28]; or do so uniformly but with some probability of error, and due to the counting argument above, using species counts exponential in the space used by the Turing machine [47]. In contrast, building on Bennett's insights relating polymer biochemistry and Turing machines [7], formal polymer systems such as Computational Nucleic Acids [31], the Biochemical Ground Form [13], DNA stack machines [42,33], DNA Turing machines [55], DNA register machines [51], and Surface CRNs [43], can all simulate classical Turing machines with no chance of error and using the same amount of space as the Turing machine.

We focus in this paper on verification of polymer systems. Specifically, we focus on the problem of, given an abstract description of a polymer system and a physical system, does the physical system "do the same thing" as the abstract description? For example, we might compare the abstract description of the DNA stack machine to its actual physical implementation [42], and wonder if the properties of a stack machine are preserved. An analogous problem came up in the finite CRN case, where verification methods (based on serializability analysis [34], pathway decomposition [46], and CRN bisimulation [27]) found subtle errors in some of the proposed CRN compilation schemes. Each of those methods has advantages and disadvantages relative to the others, but all are capable of proving relevant correspondences between the behavior of physical CRN implementations and the abstract CRNs, or pointing out implementations that fail to correspond to the abstract CRNs in important ways. For example, our previous work on CRN bisimulation [27] discusses a method of "interpreting" the chemical species of the physical system as combinations of species of the abstract system, then asking if the possible qualitative behaviors of the two systems are equivalent up to that interpretation. Two features of this theory—basing it on *weak* bisimulation where the implementation may involve multiple "silent" steps for each step of the intended formal behavior, and allowing that interpretations are one-species-to-multiple-species rather than one-to-one—are necessary for treating the implementation of CRNs using dynamic DNA nanotechnology [48,9,17,49], but result in the NP-completeness and PSPACE-completeness of several natural questions relating to finding and verifying bisimulation arguments [27]. (Methods more closely aligned with *strong* bisimulation, which in a sense require one-to-one steps and one-to-one interpretations, allow for much more efficient algorithms to verify bulk deterministic behavior and finite-volume stochastic behavior [10–12], but unfortunately these methods are not applicable to the prevailing DNA nanotechnology implementations. A variety of "equivalence" notions exist for process algebras [53]; even weaker notions such as trace equivalence can fail to respect key aspects such as the branching structure of a process or the pitfalls of deadlock. Some examples of this failure in CRNs are discussed in our work on CRN bisimulation [27].)

In this paper, we show how CRN bisimulation can be adapted to polymer CRN-like systems and can help design practical systems. In Section 2 we define a model of "linear Polymer Reaction Networks", henceforth referred to simply as PRNs, that may be considered a special case of CRNs with (usually) infinitely many species and reactions. The PRN model covers a wide range of polymer system behaviors while still being convenient for discussion of bisimulation. This model is based on species being arbitrary strings over a finite set of "monomers", and a finite set of "reaction schemata" with wildcards from

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

which concrete reactions can be enumerated. Because PRNs are a special case of infinite CRNs, and most of the theorems of CRN bisimulation do not require the CRNs to be finite, CRN bisimulation can with a few new concepts be adapted to PRNs, resulting in a new theory for PRN bisimulation. Thus, PRN bisimulation can be used to verify designs for physical implementations of polymer systems, which we show in Section 3 by proving correct an updated version of the DNA stack machine by Qian et al. [42]. Although many of the theorems (such as transitivity and modularity) from our previous work on CRN bisimulation [27] still apply to PRN bisimulation, the algorithms for finding or checking a bisimulation interpretation assume finite CRNs, and in Section 4 we show that the corresponding problems are undecidable for polymer systems.

We believe that CRN and PRN bisimulation are not just useful for verifying systems once designed, but can be used as "goalposts" to help guide the design of CRN and PRN implementations. For example, a proof by bisimulation that a certain small class of reaction mechanisms is sufficient to implement any of a larger class of reactions, suggests a design strategy involving implementing that small class of reactions. In Section 5 we give an example of such a proof that any of a class of "single-locus reaction schemata", which capture a notion of physically realistic single-step reactions, can be implemented up to PRN bisimulation by a specific set of five reaction primitives. Finally, since our linear PRNs are only one of many reasonable models of a polymer CRN-like system, in Section 6 we show how PRN bisimulation might be defined for other such systems, and suggest how our theorems might be extended accordingly.

2. Definitions

2.1. Multisets, languages, regular expressions, and automata

$\mathbb{N}$ is the set of natural numbers, $\{0, 1, 2, \dots\}$. Where A is a set, $\mathbb{N}^A$ is the set of multisets of elements of A, or equivalently, the set of functions from A to $\mathbb{N}$. Where $S \in \mathbb{N}^A$, $X \in A$ we write $S(X)$ for the count of X in S; this is consistent with S as a function $A \to \mathbb{N}$. Addition and scalar multiplication of multisets are defined componentwise. Comparison is also defined, $S \geq T$ means $\forall_X S(X) \geq T(X)$, and $S > T$ if $S \geq T$ and $S \neq T$. As we are only concerned with finite multisets, if A is infinite we use $\mathbb{N}^A$ to mean the set of multisets S with $\sum_{X \in A} S(X) < \infty$. We use the notation $\{\!|\dots|\!\}$ for multisets, e.g. $\{\!|X, Y|\!\}$ or $\{\!|2X, Z|\!\}$.

Where Σ is a set, Σ^* is the set of strings of 0 or more elements of Σ. ε is the empty string. $\emptyset$ is the empty set. A language is a set of strings. A regular language is the set of all strings that match a regular expression for some Σ. A regular expression is either a symbol in $\Sigma \cup \{\varepsilon, \emptyset\}$ or it is a^*, ab, or $(a|b)$ where a and b are already-defined regular expressions. No strings match $\emptyset$, ε matches ε, x matches x for $x \in \Sigma$, w matches $(a|b)$ if w matches a or w matches b, w matches ab if $w = w_1 w_2$ and w_1 matches a and w_2 matches b, and w matches a^* if $w = w_1 w_2$ and w_1 matches a and w_2 matches a^*.

We use finite automata, stack machines, and Turing machines for various purposes. We generally assume familiarity with them, but give a brief description. A (nondeterministic) *finite automaton* (FA or NFA) is $M = (Q, \Sigma, \delta, q_0, F)$. Q is a set of states, Σ an alphabet, $\delta \subset (Q \times \Sigma \times Q)$ a transition relation, $q_0 \in Q$ a start state, $F \subset Q$ a set of accepting states. If δ is a function $(Q \times \Sigma) \to Q$, then the automaton is *deterministic* (DFA). The automaton *accepts* a string $w \in \Sigma^*$ if there is a sequence $q_0 w_1 q_1 \dots w_n q_n$ with $(q_{i-1}, w_i, q_i) \in \delta$ and $q_n \in F$. A language $L \subset \Sigma^*$ is the language accepted by a finite automaton if and only if it is the language that matches some regular expression, and we often use the two interchangeably [30]. We also use $L(M)$ or $L(e)$ to mean the languages of an NFA M or regular expression e respectively, so $w \in L(M)$ or $w \in L(e)$ mean w is accepted by M or matches e.

A stack machine augments a finite automaton with one or more last-in-first-out memory stacks; the input string is initialized on the first stack and if the machine halts, the contents of the first stack are considered to be the output. The stack alphabet Γ is a superset of the input/output alphabet Σ. For a K-stack machine, the transition relation $\delta \subset (Q \times \{1, \dots, K\} \times \{Push, Pop\} \times \Gamma \times Q)$ indicates an initial control state, which stack to interact with, whether to push or pop, the symbol being pushed or popped, and the next control state. In this paper we only consider deterministic stack machines, so at any given time there can be only one valid transition, or none when the machine halts. Similarly, a Turing machine augments a finite automaton with an unbounded bidirectional tape of memory, with $\Sigma \subset \Gamma$ and the input and output strings being the initial and final tape contents. Here $\delta \subset (Q \times \Gamma \times \Gamma \times \{L, R\} \times Q)$ specifies the initial head state, reads a symbol, writes a symbol, moves left or right on the tape, and enters a new head state. Again, here we only deal with deterministic Turing machines, where δ is a function $(Q \times \Gamma) \to (\Gamma \times \{L, R\} \times Q)$.

2.2. Chemical reaction networks

Definition 2.1. A *Chemical Reaction Network* (CRN) is a pair $(\mathcal{S}, \mathcal{R})$, where $\mathcal{S}$ is a set of species and $\mathcal{R} \subset \mathbb{N}^{\mathcal{S}} \times \mathbb{N}^{\mathcal{S}}$ is a set of reactions.

We often use chemical reaction notation to write reactions: $(R, P) = R \to P$. If (R, P) and (P, R) are both reactions, we write $R \rightleftharpoons P$. Consistent with chemical reaction notation, when unambiguous we often identify each species A with the multiset $\{\!|A|\!\}$, so e.g. $A + B$ and $\{\!|A, B|\!\}$ refer to the same object. In general CRNs, each reaction is given a positive real number as a "rate constant", so a reaction is a triple (R, P, k), sometimes written as $R \xrightarrow{k} P$. These rate constants affect the amount each reaction happens in a given time interval and, in the stochastic model, the likelihood of a reaction happening

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

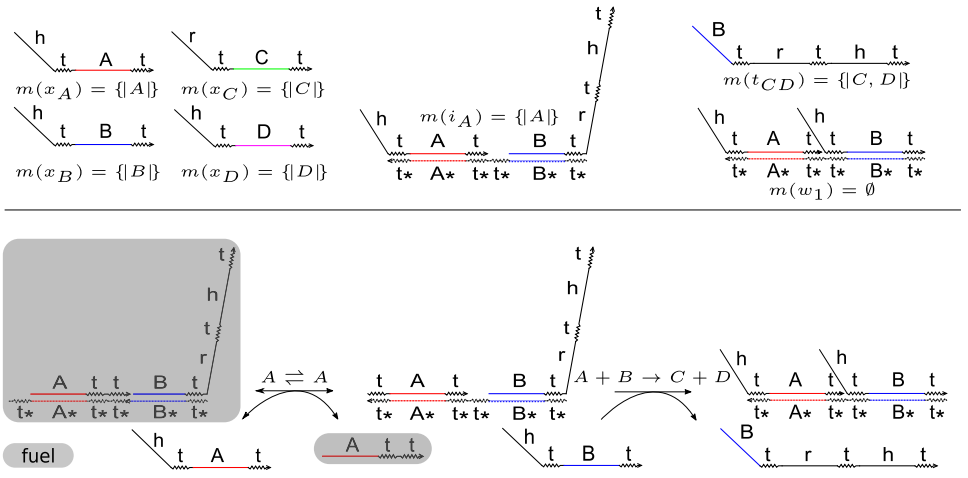


Fig. 1. An example implementation CRN with interpretation. Various DNA complexes in the implementation system are modeled as species of the implementation CRN with interpretations as multisets of formal species. (Complexes marked as "fuel" are assumed always present and are *not* modeled as species in the implementation CRN [48,27]. For example, the reaction on the left is enumerated as $x_A \rightleftharpoons i_A$, ignoring the two fuel complexes.) Interpretations of trivial (left) and nontrivial (right) reactions follow from the interpretations of the species involved. Figure adapted from [27].

relative to other reactions. The theory of CRN and PRN bisimulation deals with whether a thing *can* happen in CRNs, but not how likely it is or how much time it takes, and for those questions the values of rate constants are irrelevant (as long as they are all positive real numbers). Thus for the purposes of this paper we define reactions as pairs of reactants and products without rate constants. We further assume that no reactions $R \to P$ with $R = P$ exist.

We work with the stochastic model of CRN semantics, where a CRN starts with some count of each species present, and any possible reaction may occur, which changes the counts. Specifically, a CRN will at any point in time be at a state $S \in \mathbb{N}^{\mathcal{S}}$, a multiset of species, and for each reaction $R \to P$ where $S \geq R$ the CRN can transition from state S to state $S - R + P$. Given rate constants, this process is a continuous-time Markov chain with transition rates dependent on the rate constants and the count of reactants in S; when we only care about which transitions are possible, the previous description is equivalent to that continuous-time Markov chain. A timed trajectory is an initial state $S_0 \in \mathbb{N}^{\mathcal{S}}$ together with a (finite or infinite) sequence of reactions $r_i \in \mathcal{R}$ and times t_i at which they occur. When we care only which reactions happened in what order but not at what exact time, we can define a trajectory as an initial state followed by a sequence of reactions, without the times.

Consider a pair of CRNs $(\mathcal{S}, \mathcal{R})$ and $(\mathcal{S}', \mathcal{R}')$, where $(\mathcal{S}, \mathcal{R})$ is some abstract CRN and $(\mathcal{S}', \mathcal{R}')$ a more realistic CRN intended to implement $(\mathcal{S}, \mathcal{R})$. We call $(\mathcal{S}, \mathcal{R})$ the *formal CRN* and $(\mathcal{S}', \mathcal{R}')$ the *implementation CRN*. We previously defined a concept of *CRN bisimulation* to check whether the implementation CRN is, in fact, a correct implementation of the formal CRN [27]. CRN bisimulation is based on an *interpretation* of each implementation species as a multiset of formal species, where the implementation is correct if (for some interpretation) from any initial state the possible formal trajectories and interpreted implementation trajectories are equivalent. An example DNA implementation with interpretation is shown in Fig. 1.

Definition 2.2. An *interpretation* is a function $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ from implementation species to multisets of formal species. We extend this linearly from species to states: for $a_i \in \mathbb{N}$, $m(\sum_{i=1}^{n} a_i X_i) = \sum_{i=1}^{n} a_i m(X_i)$. We also define a *natural interpretation of reactions*: $m(R' \to P') = m(R') \to m(P')$ unless $m(R') = m(P')$, in which case $m(R' \to P') = \tau$ and we say the reaction is *trivial*. For example, if $m(i_{AB}) = A + B$, $m(x_A) = A$, and $m(t_{BC}) = B + C$ then $m(i_{AB} + x_A) = 2A + B$, and $m(i_{AB} \to x_A + t_{BC}) = A + B \to A + B + C$. Trajectories are interpreted by interpreting the initial state and each reaction.

In our previous work [27] we considered the possibility that an implementation of a reaction might have "spurious catalysts", i.e. extra species formally present in the interpretation that are not involved in or affected by the intended formal reaction. For example, in a physical DNA-based implementation, an extra strand might bind to some part of the reacting complex without affecting the actual reaction mechanism. This turns out to be a major concern in even abstract polymer systems, so we bring in that definition here.

Definition 2.3. Let $(\mathcal{S}, \mathcal{R})$ and $(\mathcal{S}', \mathcal{R}')$ be a formal and implementation CRN with $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ an interpretation of implementation species, which is extended to implementation states as in Definition 2.2. An interpretation of reactions $m_r : \mathcal{R}' \to (\mathbb{N}^{\mathcal{S}} \times \mathbb{N}^{\mathcal{S}}) \cup \{\tau\}$ is *consistent with* m if, for each $R' \to P' \in \mathcal{R}'$:

(i) If $m(R') = m(P')$ then $m_r(R' \to P') = \tau$, and
(ii) If $m(R') \neq m(P')$ then $m_r(R' \to P') = R \to P$ for some $R, P, C \in \mathbb{N}^{\mathcal{S}}$ with $m(R') = R + C$ and $m(P') = P + C$.

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

Naturally, the natural interpretation of reactions given an interpretation m is in fact consistent with m. In general below we abuse notation and use m to refer to all of the interpretation of species, the extension to states, and the chosen interpretation of reactions (natural or otherwise) consistent with the interpretation of species.

We defined correctness of an interpretation in three ways: trajectory equivalence, three conditions, and weak bisimulation. Loosely, trajectory equivalence is what we want "correctness" to imply, the three conditions are easy to define and check, and weak bisimulation is the well-studied theory of which CRN bisimulation is an instance. A theorem from our previous work proves that these three definitions are equivalent, as desired. As notation, $S \xrightarrow{r}$ means reaction r can occur in state S, while $S \xrightarrow{r} T$ means that reaction r takes state S to state T. In the implementation CRN, $S' \xRightarrow{r'}$ and $S' \xRightarrow{r'} T'$ mean the same for a sequence of zero or more trivial reactions followed by r'. Where S' is an implementation state and r is a formal reaction, $S' \xrightarrow{r} T'$ means "$S' \xrightarrow{r'} T'$ for some r' with $m(r') = r$", and similarly for $S' \xRightarrow{r} T'$, $S' \xrightarrow{r}$, and $S' \xRightarrow{r}$. In the formal CRN, $S \xRightarrow{r} T$ is equivalent to $S \xrightarrow{r} T$.

Definition 2.4 *(Three notions of correctness).* An implementation CRN $(\mathcal{S}', \mathcal{R}')$ is a correct implementation of a formal CRN $(\mathcal{S}, \mathcal{R})$ if a correct interpretation exists. An interpretation $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ is correct, in which case we say m *is a CRN bisimulation*, if any of the following three sets of conditions are true:

I **Equivalence of trajectories**
- (i) The set of formal trajectories and interpretations of implementation trajectories are equal.
- (ii) For every implementation state S', the set of formal trajectories starting from $m(S')$ and interpretations of implementation trajectories starting from S' are equal.

II **Three conditions on the interpretation**
- (i) *Atomic condition:* For every formal species A, there exists an implementation species x_A such that $m(x_A) = \{\!|A|\!\}$.
- (ii) *Delimiting condition:* The interpretation of any implementation reaction is either trivial or a valid formal reaction.
- (iii) *Permissive condition:* If $S \xrightarrow{r}$ and $m(S') = S$, there exists an implementation reaction r' such that $m(r') = r$ and $S' \xRightarrow{r'}$.

III **Weak bisimulation**
- (i) For all implementation states S':
  if $S' \xrightarrow{r} T'$, then $S \xRightarrow{r} T$ where $S = m(S')$ and $T = m(T')$.
- (ii) For all formal states S, there exists S' with $m(S') = S$, and for all such S':
  if $S \xrightarrow{r} T$, then for some T', $S' \xRightarrow{r} T'$ and $m(T') = T$.

Our previous work proved a number of theorems about CRN bisimulation. For this work, the relevant ones are those that do not assume the CRNs involved are finite. In particular, the equivalence of the three definitions of correctness, the transitivity lemma, and the modularity condition will all apply to polymer systems.

Theorem 2.1. *The three definitions of correctness, namely trajectory equivalence, the three conditions on the interpretation, and weak bisimulation, are equivalent.*

Lemma 2.1 *(Transitivity). If m_2 is a bisimulation from $(\mathcal{S}'', \mathcal{R}'')$ to $(\mathcal{S}', \mathcal{R}')$ and m_1 is a bisimulation from $(\mathcal{S}', \mathcal{R}')$ to $(\mathcal{S}, \mathcal{R})$, then $m = m_1 \circ m_2$ is a bisimulation from $(\mathcal{S}'', \mathcal{R}'')$ to $(\mathcal{S}, \mathcal{R})$.*

It is a convenient abuse of notation to write $m_1 \circ m_2$ when m_2 takes $\mathcal{S}'' \to \mathbb{N}^{\mathcal{S}'}$ and m_1 takes $\mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$. Intuitively, we extend m_1 to an interpretation on multisets over $\mathcal{S}'$; formally, for $x \in \mathcal{S}''$, $m(x) = \sum_{y \in \mathcal{S}'} m_2(x)(y) m_1(y)$, where $m_2(x)(y)$ means the count of y in $m_2(x)$ and is applied as a scalar multiplier for the multiset $m_1(y)$.

Definition 2.5 *(Modularity condition).* Let m be a bisimulation from $(\mathcal{S}', \mathcal{R}')$ to $(\mathcal{S}, \mathcal{R})$. Let $\mathcal{S}'_0 \subset \mathcal{S}'$ and $\mathcal{S}_0 \subset \mathcal{S}$ be subsets of implementation and formal species, respectively, where $y \in \mathcal{S}'_0 \Rightarrow m(y) \subset \mathcal{S}_0$. We say that m is a *modular interpretation* with respect to the *common (implementation and formal) species* $\mathcal{S}'_0$ and $\mathcal{S}_0$ if for any $x \in \mathcal{S}'$ there is a sequence of trivial reactions $\{\!|x|\!\} \xRightarrow{\tau} Y + Z$ where $Y \subset \mathcal{S}'_0$ and $m(Z) \cap \mathcal{S}_0 = \emptyset$, i.e. all common formal species in the interpretation of x are extracted as common implementation species.

Theorem 2.2 *(Modularity). Let m_1 be a bisimulation from $(\mathcal{S}'_1, \mathcal{R}'_1)$ to $(\mathcal{S}_1, \mathcal{R}_1)$ and m_2 be a bisimulation from $(\mathcal{S}'_2, \mathcal{R}'_2)$ to $(\mathcal{S}_2, \mathcal{R}_2)$ where m_1 and m_2 agree on $\mathcal{S}'_1 \cap \mathcal{S}'_2$. Let $\mathcal{S}' = \mathcal{S}'_1 \cup \mathcal{S}'_2$, $\mathcal{R}' = \mathcal{R}'_1 \cup \mathcal{R}'_2$, $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$, and $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$, and $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ equal m_1 on $\mathcal{S}'_1$ and m_2 on $\mathcal{S}'_2$. If m_1 and m_2 are both respectively modular bisimulations with respect to the common implementation species $\mathcal{S}'_1 \cap \mathcal{S}'_2$ and common formal species $\mathcal{S}_1 \cap \mathcal{S}_2$, then m is a bisimulation from $(\mathcal{S}', \mathcal{R}')$ to $(\mathcal{S}, \mathcal{R})$, and m is also modular with respect to $\mathcal{S}'_1 \cap \mathcal{S}'_2$ and $\mathcal{S}_1 \cap \mathcal{S}_2$.*

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

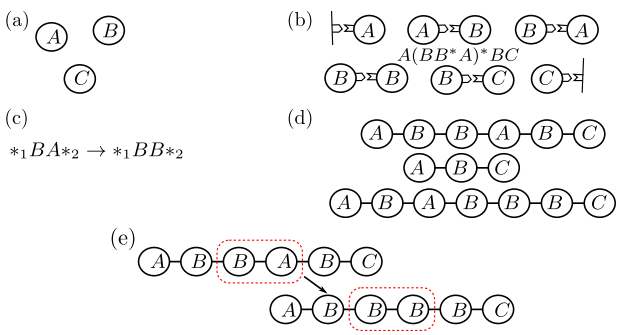


Fig. 2. A PRN is defined as a set of monomers (a), a regular expression restriction (which may be given as a compatibility relation, b), and a set of reaction schemata (one shown in c). The species of a PRN are the set of all strings of monomers that match the regular expression (three examples shown in d). Reactions are obtained by substituting strings of monomers for wildcards in the reaction schemata such that both sides of the reaction respect the compatibility relation; for example, $*_1 = AB$ and $*_2 = BC$ in $*_1BA*_2 \rightarrow *_1BB*_2$ enumerates the reaction $ABBABC \rightarrow ABBBBC$ (e).

2.3. Polymer reaction networks

A Polymer Reaction Network, like a Chemical Reaction Network, will be a set of species and a set of reactions. Unlike a typical CRN, a typical polymer system allows arbitrarily long polymers to be made from its set of monomers, and allows the same "reactions" to occur among monomers regardless of the content of the rest of the polymer. When modeled as a CRN, such a system would typically have an infinite number of both species and reactions. To handle this with a finite specification, we define a Polymer Reaction Network as a finite *species schema* and a finite set of *reaction schemata*, which will then generate the set of species and reactions.

The species of a polymer system are, in general, arbitrarily long polymers made up of a finite set of monomers. While polymers with branches and loops can exist, we wished to avoid the associated complexity. As many of the essential features we wish to study arise in linear polymer systems, we focus on those. We discuss further the reasons for this in Section 6. Thus our species will be strings over some finite "alphabet" or set of monomers. We assume that "left" and "right" are distinguished, so that the strings e.g. *abc* and *cba* are different molecules; a *b* monomer in this example would have two distinct binding sites, and the molecules differ in which site is bound to *a* and which to *c*. Again Section 6 contains discussion on what can happen if this assumption is not true.

In a physical system, typically not every string of monomers can actually exist as a polymer; some pairs of monomers will have the appropriate interfaces for binding to each other, and other pairs will not. Similarly, we assume that only some monomers can occur on the left edge of a polymer, which we represent by $\vdash$, and similarly for the right edge $\dashv$; some monomers might not be stable when unbound. We model this by letting the set of species to be restricted to those that match a specified regular expression. We justify this by showing that a more physically meaningful restriction, of only allowing certain monomers to bind to each other and to the edges, is equivalent to a regular expression up to interpretation.

One might ask why it is necessary to restrict the set of possible polymers at all. To answer that, intrinsic to our notion of CRN bisimulation is that the behaviors of the two CRNs are equivalent from *any* initial state, and we would like to have the same guarantee for PRNs. Many systems would have some polymers that can never exist physically, but if they could exist, would have absurd behavior that breaks the system. Either the regular expression restriction or the local compatibility restriction (as defined below) can solve this problem.

Given an infinite set of species generated from a finite set of monomers, we would like to specify the possible reactions of those species with a finite set of rules. A reasonable assumption is that there are a finite number of "reaction mechanisms", each of which depends on some features of its context but may be independent of others. To use an example from the stack machine of Qian et al. [42], a 0_2 (symbol 0 on stack 2) unit at the right end of a polymer can react with a query strand Q_2, removing the 0_2 symbol while leaving the rest of that polymer unchanged, and releasing a signal strand, which we call 0_2^f. This reaction depends on the 0_2 symbol being on the end of its polymer, but is independent of what else makes up the remainder of the polymer. We could write this reaction mechanism as $*_10_2 + Q_2 \rightarrow *_1 + 0_2^f$, where the string $*_10_2$ means "any polymer that ends in 0_2". Here $*_1$ is a *wildcard*, which can be filled in by any string, provided that the same wildcard is replaced by the same string in each of its appearances; since there are infinitely many possible strings that can replace $*_1$, this *reaction schema* generates infinitely many reactions. So for example, $\lambda_2 1_2 0_2 1_2 0_2 + Q_2 \rightarrow \lambda_2 1_2 0_2 1_2 + 0_2^f$ would match this schema, but $\lambda_2 1_2 0_2 + Q_2 \rightarrow \lambda_2 + 0_2^f$ would not. Other mechanisms can also be described with wildcards: $*_1 + P \rightarrow *_1 + *_1 + P$ models P catalytically copying an arbitrary string, for example, while $*_1AB*_2 \rightarrow *_1*_2$ models AB removing itself from anywhere in a polymer. We thus define the reactions of a PRN by a set of *reaction schemata*, each of which is a reaction over strings including wildcards, and generate the reactions of a PRN by substitution into the wildcards of the schemata. A pictorial example is given in Fig. 2.

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

As is usual in CRN notation, we will write $R \rightleftharpoons P$ to represent the pair of reaction schemata $R \to P$ and $P \to R$. This is valid if every wildcard that appears in either R or P appears in both, and if so, observe that the schema is truly reversible: any reaction enumerated by one direction will have its reverse reaction enumerated by the other.

Definition 2.6. A *Polymer Reaction Network* is a triple (Σ, e, Ψ) of a monomer set Σ, regular expression e over Σ, and set of reaction schemata Ψ. When the reaction schemata are irrelevant, we refer to the pair (Σ, e) as a *species schema*. A *reaction schema* $\psi \in \Psi$ is a pair (R, P) of multisets of strings over $\Sigma \cup *_{\mathbb{N}}$ where $*_{\mathbb{N}} = \{*_1, *_2, \dots\}$, such that any $*_n$ that appears in either R or P appears at least once in R. Given a PRN, it induces an *enumerated CRN* $(\mathcal{S}, \mathcal{R})$. (We sometimes write $\mathcal{S}(\Sigma, e)$, $\mathcal{R}(\Sigma, e, \Psi)$.) $\mathcal{S}$ is the set of all nonempty strings over Σ that match e. To enumerate $\mathcal{R}$, consider a reaction schema $\psi = (R, P) \in \Psi$, and for each $*_n$ that appears in ψ, choose a string s_n and substitute s_n for each appearance of $*_n$, to obtain a pair of multisets of strings over Σ. If every string obtained this way (in both R and P) matches e, then the pair of multisets is a reaction of species in $\mathcal{S}$; $\mathcal{R}$ is the set of all such reactions.

Definition 2.7. An *augmented PRN* is a triple (Σ, e, Ψ) of a monomer set Σ, regular expression e over Σ, and set of augmented reaction schemata Ψ. An *augmented reaction schema* $\psi \in \Psi$ is a reaction schema (R, P) together with, for each $*_n$ that appears in the schema, a regular expression e_n over Σ. The enumerated CRN $(\mathcal{S}, \mathcal{R})$ has $\mathcal{S}$ enumerated as usual, while $\mathcal{R}$ is the set of reactions enumerated as for an unaugmented PRN with the additional restriction that in a schema ψ, each string s_n substituted for $*_n$ must match e_n.

We do not discuss augmented PRNs much until Section 5, where among other results we show that an augmented PRN can be implemented, up to PRN bisimulation, by an unaugmented PRN.

Consider a particular type of mechanism that restricts which strings over Σ are valid polymers: only some pairs of monomers have the complementary binding sites necessary to bind. We might also assume that only some monomers are stable on the left edge of a polymer, and only some monomers are stable on the right. We can write this as a relation ρ on monomers, where $a\rho b$ means ab can bind in that order in a polymer. Technically, we must expand the domain of the relation to indicate which monomers can occur on the left and right ends of a polymer. Such a relation cannot be more powerful than a regular expression, and up to an interpretation (as defined below), we show that it is as powerful as a regular expression.

Definition 2.8. Given a monomer set Σ with notation $\Sigma_{\vdash} = \Sigma \cup \{\vdash\}$, $\Sigma_{\dashv} = \Sigma \cup \{\dashv\}$, where $\Sigma \cap \{\vdash, \dashv\} = \emptyset$, a *compatibility relation* is a relation $\rho \subset \Sigma_{\vdash} \times \Sigma_{\dashv}$. Given a monomer set Σ and compatibility relation ρ, the set of enumerated species $\mathcal{S}(\Sigma, \rho)$ is the set of all $w = x_1 \dots x_n \in \Sigma^*$ such that all of $\vdash \rho x_1$, $x_i \rho x_{i+1}$, and $x_n \rho \dashv$. As we show below that any compatibility relation can be described by a regular expression, a PRN (augmented or not) can be given as (Σ, ρ, Ψ) instead of (Σ, e, Ψ).

Lemma 2.2. *For any regular expression e over an alphabet Σ, there is a monomer set Σ', compatibility relation ρ', and interpretation $\pi : \Sigma' \to \Sigma$ such that a string $x = x_1 \dots x_n \in \Sigma^* \in L(e)$ if and only if there is a species $x' = x'_1 \dots x'_n \in \mathcal{S}(\Sigma', \rho')$ with $\pi(x'_i) = x_i$ for $1 \le i \le n$. This construction can be done with $\Sigma' = \Sigma$, $\forall_x \pi(x) = x$ if and only if e has the property that for $x \in \Sigma$, $u_1, v_1, u_2, v_2 \in \Sigma^*$, if $u_1 x v_1$ and $u_2 x v_2$ match e then so does $u_1 x v_2$; if so, we say that e is* local. *Conversely, given any monomer set Σ with compatibility relation ρ there is a local regular expression e with $\mathcal{S}(\Sigma, e) = \mathcal{S}(\Sigma, \rho)$. All of these transformations can be computed in polynomial time.*

Proof. Consider a nondeterministic finite automaton M that recognizes strings that match e. Where Q is the set of states of M, let $\Sigma' = \Sigma \times Q$ and let $\pi((x, q)) = x$. Let $((x_1, q_1), (x_2, q_2)) \in \rho'$ if and only if M can transition from state q_1 to state q_2 by reading x_2, let $(\vdash, (x, q)) \in \rho'$ if and only if M can transition from its start state q_0 to q by reading x, and $((x, q), \dashv) \in \rho'$ if and only if q is an accept state of M. Valid polymers correspond exactly to accepting computation paths of M on their interpretations.

If e is local, then for any states q_i, q_j both of which have incoming transitions labeled x, either one of those transitions is not reachable on any string and can be removed, or given locality, the set of strings accepted after reaching q_i and q_j are the same. In that case, an equivalent automaton has q_i and q_j collapsed into one state. Repeating this process constructs a finite automaton that recognizes e where for each state x there is at most one state q_x with incoming transitions labeled x (it may be that $q_x = q_y$ for $x \neq y$). Applying the above construction to this new finite automaton and labeling the monomer (x, q_x) as x gives the desired (Σ, ρ).

Given (Σ, ρ), a finite automaton as above can be easily constructed: for each monomer x a state q_x, with $q_x \xrightarrow{y} q_y \iff x\rho y$, $q_0 \xrightarrow{x} q_x \iff \vdash \rho x$, and q_x accepts if and only if $x\rho\dashv$. As above, valid polymers correspond exactly to accepting computation paths of M. □

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

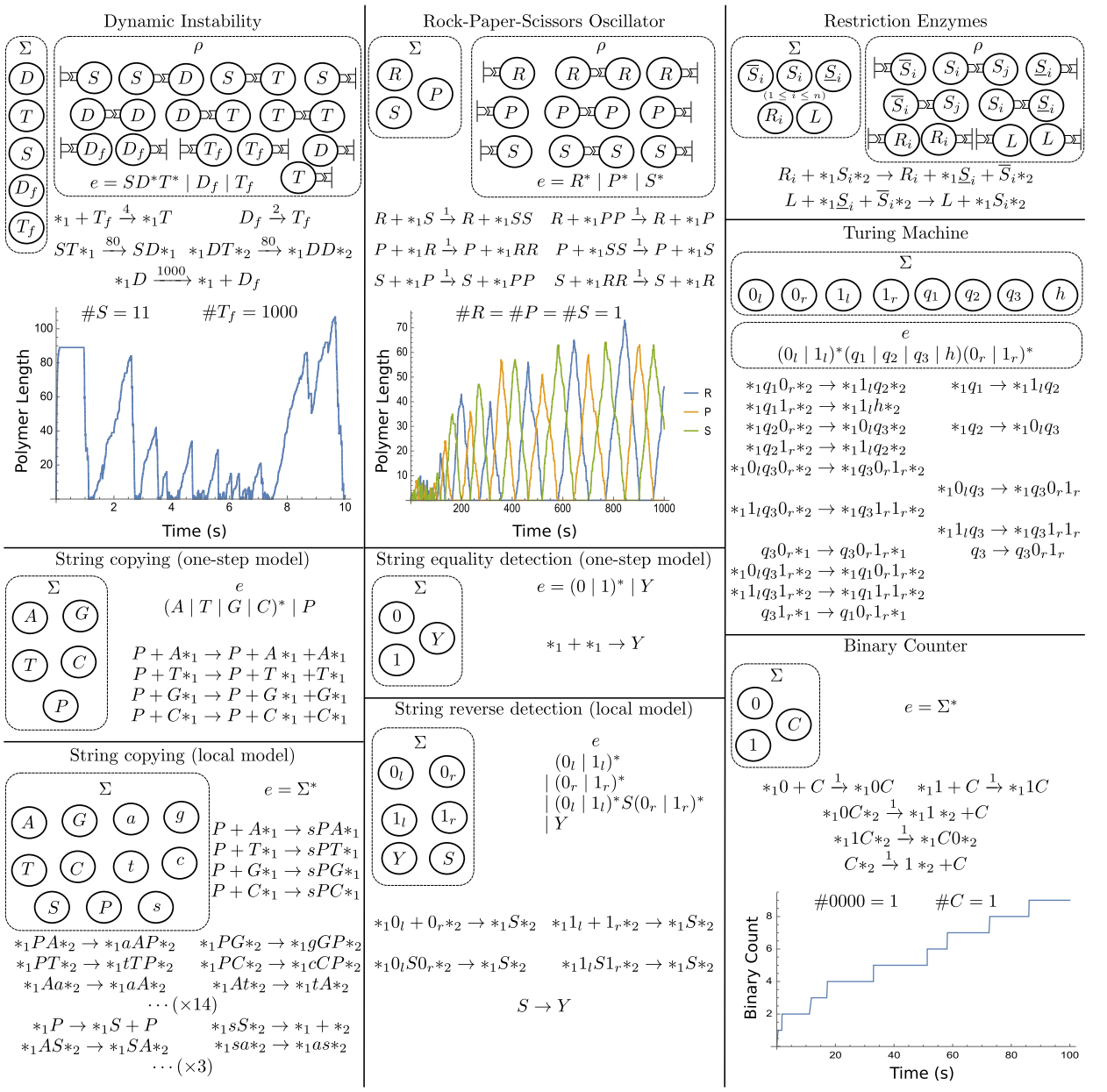


Fig. 3. These example Polymer Reaction Networks demonstrate various features of the PRN model. Shown for each PRN is the monomer set Σ; the regular expression e describing the set of polymers and/or, if e is local, the equivalent compatibility relation ρ; and the reaction schemata. Additionally for some PRNs, rate constants are assigned to each reaction schema and a sample stochastic simulation, as described above, from the given initial conditions is shown. Which of these PRNs are useful and/or interesting is left as an exercise to the reader.

2.4. PRN examples

To help get a feel for the PRN model, we provide several illustrative examples of PRNs and their behaviors in Fig. 3. Three of the examples are explicitly simulated, and in these cases we provide rate constants for each of the reaction schemata. It is important to note that some aspects of PRN behaviors may be sensitive to the rate constants (such as whether it is likely to reach some state from another state) while other aspects will be rate-independent (such as whether it is possible to reach some state from another state). CRN bisimulation and thus PRN bisimulation are rate-independent theories, and thus they can only speak to that aspect of the behavior. The guarantees they provide are stronger than just reachability; they also guarantee a correspondence between the trajectories – i.e. *how* one gets there. For example, bisimulation will distinguish between a clockwise and a counter-clockwise trajectory through a cycle, while analysis based purely on reachability would not; on the other hand, bisimulation would not distinguish between a clockwise trajectory that likely spends most of its

time in one state and rushes through the others, versus a clockwise trajectory that distributes its time roughly uniformly. It is a useful exercise to consider which behavioral features of the examples in Fig. 3 are rate-independent, and thus likely to be captured by PRN bisimulation.

Because a PRN is an infinite CRN, we can simulate a PRN using the semantics of a stochastic CRN, such as Gillespie's algorithm [23]. Although the number of reactions in a PRN may be infinite, if at a given time there are a finite number of polymers each of finite length, then at that time there will be a finite number of reactions possible. The only difficulty is that the set of possible reactions in general must be re-computed every time a new species is produced, preventing various methods of optimizing the Gillespie algorithm that are possible for a finite CRN. However, implementing the basic Gillespie algorithm is possible using a "just-in-time" reaction enumeration approach, which has been done in Visual DSD for polymer-like systems created from DNA strand displacement networks [35]. A second issue that must be addressed is how to assign rate constants to each specific CRN reaction that is derived from the PRN schemata. Here, we assume that each reaction schema has a rate constant, that every reaction enumerated from that schema has that schema's rate constant, and that the same reaction enumerated multiple times (from different schemata and/or different substitutions into the same schema) has as its rate constants the sum of schema rate constants from all of its enumerations (which is guaranteed to be finite). Other methods of specifying rate constants are possible.

Fig. 3 gives examples that showcase various relevant features of the Polymer Reaction Network model. Code for simulating all these systems can be found in our Mathematica package [54]. We discuss some of those examples in further detail here.

Example 2.1 *(Dynamic instability).* Shown in the upper left of Fig. 3, this PRN has 5 monomers, $\Sigma = \{D, T, S, D_f, T_f\}$; regular expression restriction $SD^*T^* \mid D_f \mid T_f$, which is equivalent to the compatibility relation ρ shown; and 5 reaction schemata.

We give an example PRN that describes a model of dynamic instability, inspired by (but an oversimplification of) biological microtubules [18]. Our model, in English, is as follows: A polymer is a seed S followed by any number of D monomers then any number of T monomers; those latter two types of monomers can also exist free-floating in solution, represented by D_f and T_f. A free T_f monomer can attach onto the right end of a polymer as a T; T monomers can convert into D monomers starting from the left end of a polymer; and only D monomers can fall off the right end. In solution, free D_f converts back into T_f, to complete the cycle.

In the compatibility relation, first observe the patterns $(\vdash, D_f)$, $(D_f, \dashv)$ and similarly $(\vdash, T_f)$, $(T_f, \dashv)$, with no other occurrences of D_f or T_f; the result of this is that D_f and T_f can exist as monomers but can't polymerize. Effectively, D_f and T_f are CRN species. The only other presence of $\vdash$ is $(\vdash, S)$, so any polymer must start with S, and given $(S, \dashv) \in \rho$, can end immediately. Otherwise, (S, D), (D, D), (D, T), (T, T), and $(T, \dashv)$ allow for one or more D then one or more T, while (S, T) and $(D, \dashv)$ allow the possibility of 0 D's or 0 T's, respectively. Thus the set of possible polymers is, as claimed in Fig. 3, represented by the regular expression $SD^*T^* \mid D_f \mid T_f$.

The reaction schemata then correspond to the above description of the system's behavior; for example, $*_1 + T_f \to *_1 T$ is a free T_f attaching to the right edge of a polymer, while $*_1 D \to *_1 + D_f$ is a D falling off the right edge. Recall that in enumerating reactions, a string can only be substituted for a given wildcard if doing so respects ρ *in both the reactants and the products*. Thus in the reaction schema $*_1 + T_f \to *_1 T$, while for example $*_1 = D_f$ would make valid reactants $D_f + T_f$, the product $D_f T$ does not respect ρ, and $D_f + T_f \to D_f T$ is not a reaction in this PRN. (This reaction schema is the only one in this figure for which this consideration is relevant. Replacing $*_1 + T_f \to *_1 T$ with either the one schema $S *_1 + T_f \to S *_1 T$ or the three schemata $S + T_f \to ST$, $*_1 D + T_f \to *_1 DT$, and $*_1 T + T_f \to *_1 TT$ would give the same set of reactions without taking advantage of this technicality, and the generalization of this is discussed further elsewhere in this paper.)

The graph shown is from a Mathematica simulation [54] of this PRN as discussed previously, from an initial state with 11 copies of (the length-1 polymer) S and 1000 copies of T_f. Mathematica was instructed to track the length of one individual S polymer, and the plot shows that one polymer's length over time. Several phenomena are evident: growth of polymers while the cap of T remains stable, shrinkage during a catastrophe event when the cap of T dissolves, and occasional rescue.

Example 2.2 *(Copy-tolerant Turing machine).* Shown in the middle right of Fig. 3, this PRN has 8 monomers; a regular expression restriction $(0_l \mid 1_l)^*(q_1 \mid q_2 \mid q_3 \mid h)(0_r \mid 1_r)^*$; and 15 reaction schemata corresponding to the 6 transition rules of a particular 3-state Turing machine.

The PRN shown in Fig. 3 simulates a particular 3-state Busy Beaver Turing machine with transition rules shown in Table 1. This Turing machine, from state q_1 on a blank (all-0) initial tape, halts after 14 steps with 6 1's on the tape [37]. Similarly, in the PRN the polymer q_1 will, after 14 unimolecular reactions, become the polymer $1_l 1_l 1_l h 1_r 1_r 1_r$ (and any polymer with q_1 preceded by any number of 0_l and followed by any number of 0_r will have a similar trajectory).

This PRN is an instance of a general method of simulating Turing machines with linear PRNs, using unimolecular reaction schemata corresponding to the transition rules of the Turing machine; for example, the reaction schema $*_1 q_1 0_r *_2 \to *_1 1_l q_2 *_2$ corresponds to the rule "in state 1, reading 0, write 1, move right, and transition to state 2". Transition rules reading a 0 require an additional reaction schema for the right edge of the tape, assuming blank spaces are treated as

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

Table 1
The transition rules of the Turing machine whose polymer implementation is shown in Fig. 3.

	q_1	q_2	q_3
0	1, R, q_2	0, R, q_3	1, L, q_3
1	1, R, h	1, R, q_2	1, L, q_1

0, and transition rules moving left require multiple reaction schemata for a 0, 1, or left edge of the tape to the left of the current square; this causes the 6 transition rules of the 3-state, 2-symbol Turing machine to require 15 reaction schemata.

A state of a Turing machine tape is any number of tape symbols, followed by a Turing machine head state, followed by any number of tape symbols. For a 2-symbol 3-state Turing machine this can be described by the regular expression $(0 \mid 1)^*(q_1 \mid q_2 \mid q_3 \mid h)(0 \mid 1)^*$, which is not local. If we wanted to physically implement this restriction, we could not do so using only nearest-neighbor interactions. Applying the construction from Lemma 2.2 requires 0 and 1 to each have two monomers representing them, leading to the regular expression $(0_l \mid 1_l)^*(q_1 \mid q_2 \mid q_3 \mid h)(0_r \mid 1_r)^*$ shown in Fig. 3. (One could imagine some creative methods to physically implement the nonlocal regular expression, such as having a q_i monomer destabilize 0 and 1 monomers to its right and left in different ways. We would argue that such creative solutions are best modeled by treating "0 destabilized by a q on its right" and "0 destabilized by a q on its left" as distinct monomers, since they would be physically distinct and have different behaviors. This is equivalent to the $0_l, 0_r$ model shown.) As this regular expression is local, it can be implemented by a compatibility relation ρ containing pairs $(\vdash, 0_l)$, $(\vdash, 1_l)$; (a, b) for $a, b \in \{0_l, 1_l\}$; (a, q) for $a \in \{0_l, 1_l\}$, $q \in \{q_1, q_2, q_3, h\}$; (q, a) for $q \in \{q_1, q_2, q_3, h\}$, $a \in \{0_r, 1_r\}$; (a, b) for $a, b \in \{0_r, 1_r\}$; $(0_r, \dashv)$, and $(1_r, \dashv)$. The generalization to a Turing machine with any number of states and/or tape symbols is obvious.

We previously mentioned that well-mixed CRNs can simulate Turing machines with arbitrarily small probability of error but using species counts exponential in the space of the Turing machine [47], and provably cannot simulate Turing machines deterministically [2,15,36]. Formal polymer systems such as Qian et al.'s stack machine [42] and Cardelli et al.'s Biochemical Ground Form (BGF) [13] are already known to be able to simulate Turing machines deterministically, in some cases with no time or space slowdown. The PRN shown in Fig. 3, if it can be implemented, has a feature that the DNA stack machine and the BGF register machine do not: because a Turing machine tape is encoded in a single polymer and its transitions are all unimolecular reactions, multiple copies of the Turing machine can coexist in the same solution without interfering with each other. Bennett's hypothetical polymer Turing machine [7] also shares this desirable feature.

Example 2.3 *(String copying).* The middle left and lower left of Fig. 3 show respectively a one-step nonlocal and a multi-step local model of string copying with PRNs. The one-step model has monomers $\Sigma = \{A, T, G, C, P\}$, local regular expression restriction $(A \mid T \mid G \mid C)^* \mid P$, and four reaction schemata to copy, in one step catalyzed by P, any string that starts with A, T, G, or C. The local model takes a string made of monomers A, T, G, C and transcribes the corresponding string of a, t, g, c, using monomers P, s, and S to copy one monomer at a time and eventually split the two strings.

The string copying PRNs illustrate an interesting feature of wildcards using models inspired by, but not accurate to the mechanisms of, DNA and RNA polymerases. The one-step model can be thought of as an abstraction of the *result* of DNA polymerase: where P exists only as a monomer and any string over A, T, G, C is possible under e, the PRN has four reaction schemata of the form $P + x*_1 \to P + x*_1 + x*_1$ for $x \in \{A, T, G, C\}$; note that given e, this implies that $*_1$ must be made of A, T, C, G. (The reaction schema $P + *_1 \to P + *_1 + *_1$ would have allowed the reaction $2P \to 3P$, which is certainly not what we wanted and is known to go to infinity in finite time [16]. If we were to use an *augmented* PRN model to constrain $*_1$ to match $(A|T|C|G)^*$, then this single reaction schema would suffice.) Because when enumerating reactions from a schema, each instance of a given wildcard is substituted by the same string, the result of these schemata is to copy any string made of A, T, G, C, catalyzed by P. However, the idea that a second copy of an arbitrarily long string can be produced in one step is not physically realistic, and while this PRN may represent the *result* of DNA polymerase, it certainly does not represent its mechanism.

The local model of string copying is a more realistic PRN that accomplishes the result of RNA polymerase, i.e. given a string over A, T, G, C it creates an additional copy of the corresponding string over a, t, g, c, catalyzed by P. (While this is a more physically realistic mechanism with a result analogous to RNA polymerase without the t vs u distinction, it is not in fact the mechanism of RNA polymerase, because that mechanism cannot be modeled with only linear polymers.) This concept of local mechanisms as "physically realistic", in a sense that many exotic uses of wildcards are not, is formalized in the concept of single-locus PRNs in Section 5.

2.5. PRN bisimulation

Because a PRN is an infinite CRN, we can extend the definition of CRN bisimulation from CRNs to PRNs, but doing so requires an infinite interpretation. To finitely express an infinite interpretation, we build an interpretation of species from an interpretation of monomers. The obvious thing to do is to have the interpretation of a polymer be the concatenation of interpretations of its monomers, but that would not allow interpreting one implementation polymer as a multiset of

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

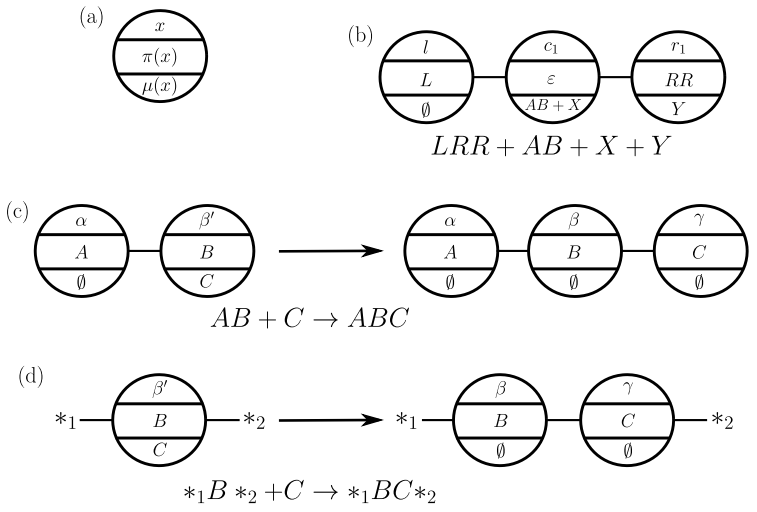


Fig. 4. Features of a polymer interpretation. An implementation monomer x has a pair of interpretations $\pi(x)$ and $\mu(x)$, shown respectively in the top, middle, and bottom parts of the node (a). An implementation polymer (e.g. lc_1r_1) is interpreted by concatenating π-interpretations and adding μ-interpretations (b). Given an interpretation of species, interpretations of states and reactions (c) follow as in CRN bisimulation [27], Definitions 2.2 and 2.3. Intuitively, interpretations of reaction schemata can follow from interpretations of monomers (d, see also Theorem 2.3); see text for details.

formal polymers (as is possible in the finite case). We therefore require that our interpretation be built from two finite functions, μ and π, defined on the implementation monomers. Here $\pi(x)$ is the contribution of the monomer x to the polymer it is contained in and $\mu(x)$ is a multiset of additional, free-floating species represented by x. We sometimes say that x *polymerizes as* $\pi(x)$ and *carries* $\mu(x)$. A convenient diagramatic notation that highlights monomer names and their interpretations, is illustrated in Fig. 4. Because in PRNs every species is thought of as a polymer, even monomers that never "polymerize", in such cases we will typically encode the interpretation in π and leave μ empty. Note that with this notion of interpretation, a single implementation monomer cannot represent ("polymerize as") monomers of two distinct arbitrary-length polymers, as would be needed if we wanted arbitrary DNA duplexes to represent the multiset containing the two strands as separate species. We refrain from attempting that generalization here.

Definition 2.9. Given a formal PRN (Σ, e, Ψ) and implementation PRN (Σ', e', Ψ'), a *polymer interpretation* is a pair (π, μ) of functions $\pi : \Sigma' \to (\Sigma \cup \{+\})^*$ and $\mu : \Sigma' \to \mathbb{N}^{\mathcal{S}}$. These functions *induce* an interpretation $m : \mathcal{S}' \to \mathbb{N}^{\Sigma^*}$ defined by

$$m(x_1 \dots x_n) = \pi(x_1) \dots \pi(x_n) + \sum_{i=1}^{n} \mu(x_i).$$

As a notation, we use $\pi(x_1 \dots x_n)$ to mean $\pi(x_1) \dots \pi(x_n)$ and $\mu(x_1 \dots x_n)$ to mean $\sum_{i=1}^{n} \mu(x_i)$, so where $s = x_1 \dots x_n \in \mathcal{S}'$ the above could have read $m(s) = \pi(s) + \mu(s)$.

The symbol $+$ is interpreted as breaking a polymer, matching the notation for separate CRN species (see Fig. 4): if the π-interpretation of a polymer reads as e.g. $AB + CD$, then it is interpreted as separate species AB and CD (plus its μ-interpretation). For example, if $\pi(x) = AB + CD$ and $\mu(x) = EF + 2GH$ then $m(xx) = AB + CDAB + CD + 2EF + 4GH$. Redundant $+$ are allowed and treated as redundant. If an implementation species x carries nothing, $\mu(x) = \emptyset$, and if it polymerizes as nothing, $\pi(x) = \varepsilon$, the empty string. Note that an empty-string polymer is not considered to be a species, so if all $\pi(x_i) = \varepsilon$ (or some number of $+$ with no actual formal monomers) then $m(x_1 \dots x_n) = \sum_{i=1}^{n} \mu(x_i)$. As in CRN bisimulation it is possible for a given $m(x_1 \dots x_n) = \emptyset$, in this case if all $\pi(x_i) = \varepsilon$ and all $\mu(x_i) = \emptyset$.

The induced interpretation m maps $\mathcal{S}'$ to $\mathbb{N}^{\Sigma^*}$, multisets of strings of formal polymers, while what CRN bisimulation wants is an interpretation that maps $\mathcal{S}'$ to $\mathbb{N}^{\mathcal{S}}$, multisets of formal species (i.e., strings that match e). If we had that, we could straightforwardly adapt the definition of CRN bisimulation to PRNs, although there is one remaining snag. Consider an interpretation where $\pi(x) = X$ and $\mu(x) = Y$ while $\pi(z) = Z$ and $\mu(z) = \emptyset$, in which case the reaction scheme $*_1x*_2 \to *_1z*_2$ intuitively should be interpreted as $*_1X*_2 + Y \to *_1Z*_2$. However, substituting x for $*_1$ and ε for $*_2$ yields $xx \to xz$, which would be interpreted as $XX + 2Y \to XZ + Y$, which cannot be obtained by substituting any two strings into the given formal reaction scheme. Avoiding this requires using the spurious-catalyst extension of CRN bisimulation from [27], in which an implementation reaction whose interpretation has catalysts can be labeled as a formal reaction without some or all of those catalysts. In the default case, we assume any species present in μ of some monomer in a wildcard are spurious catalysts. The intuition here is that we expect the reaction schema to correspond to a molecular mechanism, so the monomers that are explicitly represented (i.e. not in a wildcard) are the only ones that could be playing an important

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

(e.g. catalytic) role. In more complex cases, such as ones similar to those where the spurious catalyst interpretation would be necessary in CRN bisimulation, a different definition of spurious catalysts may be necessary.

Definition 2.10. Let $r' = R' \to P' \in \mathcal{R}(\Sigma', e', \Psi')$ be an implementation reaction that can only be enumerated in one way, and $\psi' \in \Psi'$ be the reaction schema from which r' is enumerated. Let $*_1, \ldots, *_n$ be the wildcards appearing in ψ', a_i be the count of appearances of $*_i$ in the reactants of ψ' and b_i the same for the products, and $s_1, \ldots, s_n$ the substitutions that enumerate r' from ψ'. Where $C_R = \sum_{i=1}^{n} a_i \mu(s_i)$ and $C_P = \sum_{i=1}^{n} b_i \mu(s_i)$, if $C_R = C_P$ then the multiset of *wildcard-based catalysts* of r' is $C = C_R = C_P$, and the *wildcard-based interpretation* of r' is $m(r') = R' - C \to P' - C$.

If somehow two distinct implementation schemata produce identical implementation reactions that would have different spurious catalysts (for example, $xy*_1 \to xa*_1$ and $*_1yz \to *_1az$ both enumerate the reaction $xyz \to xaz$), then the above wildcard-based interpretation of that reaction would depend on which schema it was enumerated from; this is inconsistent with our formalism that does not allow CRNs to have multiple copies of identical reactions and with the requirement that the interpretation of a reaction must be a single formal reaction (or τ). Although these situations are avoided in the examples and theorems discussed in this paper, such situations could arise in some cases. For example, there may be distinct schemata that represent distinct physical mechanisms that, in some cases, happen to produce the same result. Another situation might be when schemata are generated by an algorithm that is allowed to be redundant as long as it covers all cases. In such cases, the user of the PRN bisimulation theory would have to make decisions about how to assign spurious catalysts or augment the theory.

Most of our theorems are not affected much by this edge case. Section 3 discusses a particular system in which this edge case does not occur; Section 4 discusses complexity results that are not made harder by the edge case; and theorems in Section 5 either construct a system in which the edge case doesn't occur, or assume the existence of a bisimulation interpretation which implicitly assumes that, if the edge case does occur, then the interpretation specifies how to handle it.

We can get a proper interpretation $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ in either of two ways. The obvious way is if $x \in \mathcal{S}'$ guarantees $m(x) \subset \mathcal{S}$:

Definition 2.11. Let (Σ, e) and (Σ', e') be a formal and implementation species specification, with π-interpretation $\pi : \Sigma' \to (\Sigma \cup \{+\})^*$. Introduce notation for regular expression e that $e[+] = (e|\varepsilon)(+(e|\varepsilon))^*$, i.e. a string matches $e[+]$ if it is a $+$-separated list of strings that each match e or are empty. We say that π satisfies the *compatibility condition* if $x_1 \ldots x_n$ matching e' implies $\pi(x_1) \ldots \pi(x_n) \in L(e[+])$.

In this case, given that any μ by assumption is a function $\Sigma' \to \mathbb{N}^{\mathcal{S}}$, the induced m will in fact be a CRN interpretation $\mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ as desired, and asking whether m is a CRN bisimulation is well-defined.

If π does not satisfy the compatibility condition, the structure of the implementation reaction schemata may still make the system well-behaved. For example, consider the following implementation for the Rock Paper Scissors Oscillator of Fig. 3:

$$r + *_1 s \to r + *_1 sos \qquad p + *_1 r \to p + *_1 ror \qquad s + *_1 p \to s + *_1 pop$$
$$r + *_1 pop \to r + *_1 \qquad p + *_1 sos \to p + *_1 s \qquad s + *_1 ror \to s + *_1 r$$

with the interpretation $m(r) = (R, \emptyset)$, $m(p) = (P, \emptyset)$, $m(s) = (S, \emptyset)$, $m(o) = (\varepsilon, \emptyset)$ and the species regular expression $e' = (r|p|s)(o(r|p|s))^*$ that allows o to bind to any information-bearing monomer on its right or left. This PRN does not satisfy the compatibility condition because, for example, $rop \in L(e')$ but $RP \notin L(e)$. Nonetheless, started from any implementation state whose interpretation is a valid formal state, only valid next states can be produced. For example, $sosos \in L(e')$ and $r + sosos \to r + sososos \in \mathcal{R}(\Sigma', e', \Psi')$ preserves the restriction that polymers are of homogeneous type.

We capture this concept as follows:

Definition 2.12. Let (Σ, e) be a formal species specification and (Σ', e', Ψ') an implementation PRN, with π-interpretation $\pi : \Sigma' \to (\Sigma \cup \{+\})^*$. We say that π satisfies the *consistency condition* if, for any reaction $R' \to P'$ enumerated from a schema in Ψ', if all $x \in R'$ has $\pi(x) \in L(e[+])$ then all $y \in P'$ has $\pi(y) \in L(e[+])$.

If (π, μ) is a polymer interpretation where π satisfies the consistency condition, then let $\mathcal{S}'_0 = \{x \in \mathcal{S}' \mid \pi(x) \in L(e[+])\}$. Naturally, m restricted to $\mathcal{S}'_0$ will be a function $\mathcal{S}'_0 \to \mathbb{N}^{\mathcal{S}}$; the consistency condition implies that "restricting" the enumerated implementation CRN to $\mathcal{S}'_0$ is well-defined. That is, the CRN $(\mathcal{S}'_0, \mathcal{R}'_0)$ where $\mathcal{R}'_0$ is the set of reactions with all reactants and products in $\mathcal{S}'_0$ contains every reaction with all reactants in $\mathcal{S}'_0$. Then $m : \mathcal{S}'_0 \to \mathbb{N}^{\mathcal{S}}$ is a CRN interpretation from that CRN to the enumerated formal CRN, and asking whether m is a CRN bisimulation is well-defined.

Definition 2.13. Let (Σ, e, Ψ) and (Σ', e', Ψ') be a formal and implementation PRN, with polymer interpretation (π, μ) and induced CRN interpretation m. Let $\mathcal{S} = \mathcal{S}(\Sigma, e)$ and $\mathcal{S}' = \mathcal{S}(\Sigma', e')$. We say (π, μ) is a *PRN bisimulation* if π satisfies the compatibility condition and m is a CRN bisimulation. We say (π, μ) is a *PRN bisimulation up to reachability* (from valid initial states) if π satisfies the consistency condition and m (restricted to $\mathcal{S}'_0$ as defined above) is a CRN bisimulation.

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

In our previous work we proved that CRN bisimulation was equivalent to up-to-interpretation trajectory equivalence [27], and that result holds for infinite CRNs and thus for PRNs. Because of this, we use CRN bisimulation (with either the compatibility condition or the consistency condition) as the definition of PRN bisimulation, despite the fact that the atomic, delimiting, and permissive conditions now each refer to an infinite set of objects. We could, instead, have defined similar conditions on the polymer structure of a PRN, and showed that those conditions imply the three conditions of CRN bisimulation, just as a polymer interpretation induces a CRN interpretation. Such conditions would capture most of the typical PRN implementations, while missing some edge cases that nevertheless satisfy CRN bisimulation. Although not the definition of PRN bisimulation, one such set of sufficient conditions is useful for proving that common implementations satisfy PRN bisimulation.

Theorem 2.3. *Let (Σ, e, Ψ) and (Σ', e', Ψ') be a formal and implementation PRN with polymer interpretation (π, μ). Assume either π satisfies the compatibility condition and m is the induced CRN interpretation, or the system satisfies the consistency condition and m is the CRN interpretation restricted to formally valid species and reactions. If (π, μ) satisfies the following three conditions, then m is a CRN bisimulation (and thus a PRN bisimulation or PRN bisimulation up to reachability depending on whether it satisfies the compatibility or consistency condition):*

1. Polymer Atomic Condition*: For each formal monomer X there is a canonical implementation monomer x_0 with $\pi(x_0) = X$ and $\mu(x_0) = \emptyset$. e and e' are local and equivalent to compatibility relations ρ and ρ' respectively, where for all formal monomers X, Y with canonical implementation monomers x_0, y_0 respectively, $(X, Y) \in \rho \Rightarrow (x_0, y_0) \in \rho'$ (also $X = \vdash = x_0$ or $Y = \dashv = y_0$).*
2. Polymer Delimiting Condition*: For each reaction schema in the implementation PRN, each wildcard appears the same number of times in the products as it does in the reactants, and syntactically replacing each non-wildcard monomer with its π and μ either produces equal expressions for the reactants and products or produces a formal reaction schema.*
3. *m as a CRN interpretation satisfies the permissive condition.*

Proof. The polymer atomic condition implies the atomic condition: any formal polymer can be built up from its corresponding implementation monomers. The polymer delimiting condition implies the delimiting condition: any implementation reaction enumerated from a schema will be interpreted as trivial or as a formal reaction enumerated from the "syntactically interpreted" formal reaction schema. (This last statement requires either the compatibility condition to imply that the resulting formal reaction is valid, or the consistency condition for an implementation reaction in the restricted subsystem to imply the same.) □

Note that the above conditions are sufficient, but not necessary, for PRN bisimulation. In some sense they describe a "natural" or "polymer" way to satisfy the conditions of PRN bisimulation. However, a pair of PRNs with a compatible or consistent interpretation may happen to satisfy the conditions of CRN bisimulation, and thus PRN bisimulation, without satisfying the stronger polymer conditions.

3. Verifying the DNA stack machine

We show the use of the Polymer Reaction Network model, and PRN bisimulation, by analyzing an existing DNA strand displacement system that uses polymers. Specifically, we analyze the system proposed by Qian et al. to implement arbitrary stack machines using DNA strand displacement [42]. This system uses a reversible addition primitive to add units representing stack symbols onto a growing stack, and uses a systematic CRN implementation for state transitions. The reversible addition primitive can grow polymers of unbounded length (as desired for a stack machine), and thus the system cannot be modeled as a Chemical Reaction Network. Modeling the DNA stack machine as a Polymer Reaction Network allows us to check whether the strand displacement system is a correct bisimulation of an abstract stack machine. We show that the obvious interpretation on the DNA stack machine, with a correction for irreversible reactions, is a bisimulation between the DNA strand displacement system and the set of abstract reactions discussed in the original stack machine paper.

To prove that two systems are (or are not) equivalent using PRN bisimulation, we need to find an interpretation (or consider all potential interpretations for the negative case), check the compatibility or consistency condition (the stack machine as we model it will satisfy the compatibility condition), then check the atomic, delimiting, and permissive conditions. All of that assumes the two systems are a formal and implementation PRN; if not, we need to model each system as a linear PRN. For the stack machine, the formal system is a linear PRN and the implementation a DSD system; we use reaction enumeration as described below to describe it as an implementation PRN. To take advantage of the modularity condition from our previous work [27], we will add an additional step of dividing both systems into modules before checking the three conditions of CRN bisimulation. Thus the steps are as follows: enumerate the reaction schemata of the DSD system as an implementation PRN; construct an interpretation; check the compatibility condition; modularize; check the atomic, delimiting, and permissive conditions for each module.

When enumerating a DSD system without polymers as a CRN, every new DNA complex is a new species in the CRN. With polymers, on the other hand, most DNA complexes are polymers made out of monomer subunits; this requires identifying which patterns of DNA strands are the formal monomers. The naive approach, of having each strand be a monomer, might

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

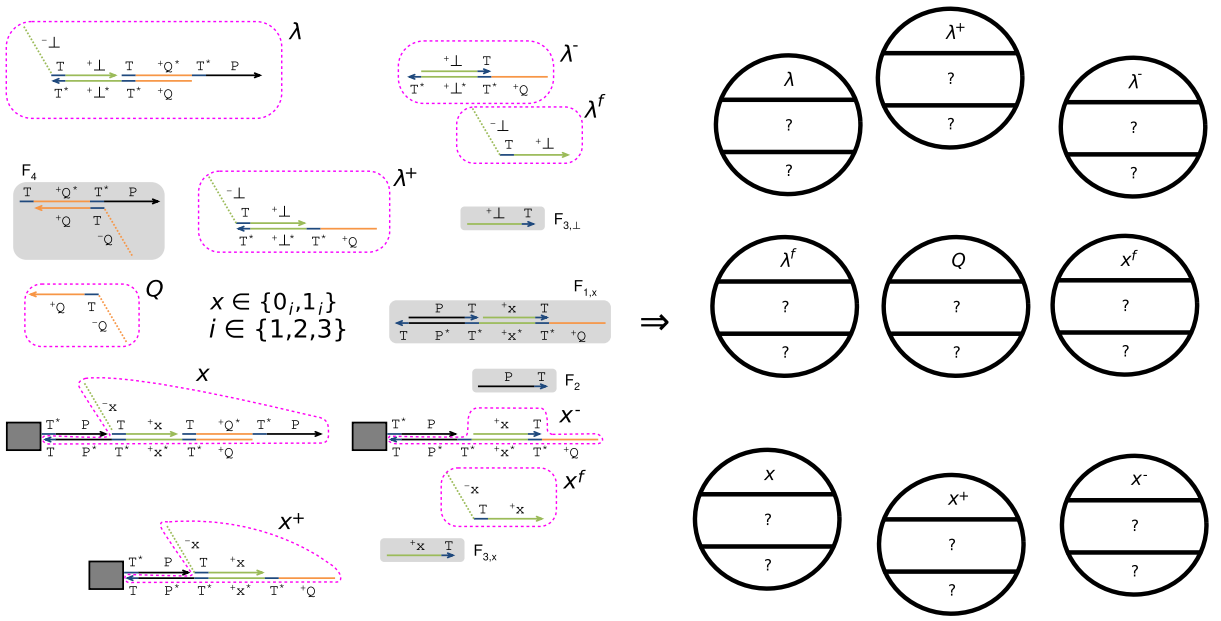


Fig. 5. Choice of monomers in Qian et al.'s DNA stack machine [42]. The π and μ interpretations are shown as "?" because our next challenge will to figure out what they should be.

work in a model that explicitly permits branched polymers, but fails in our linear polymer model given strands with more than two binding domains. We do not currently have a way to automate the identification of monomer subunits in polymer DSD systems. Therefore we describe below (and in Fig. 5) which DNA complexes we choose as monomers in Σ', after which e' is a local regular expression generated from a ρ' where monomer complexes can bind if they have complementary long domains and Ψ' is determined by the enumerated set of strand displacement reactions. Although currently Peppercorn [5] cannot automatically enumerate polymer reaction schemata, below we give an implementation PRN that we claim is the result of applying Peppercorn's condensed semantics to the DSD stack machine [42], and invite the reader to confirm that this is the case. For the six-state three-stack machine in Figure 4(a) of Qian et al. [42], the resulting implementation PRN is (Σ', e', Ψ') as follows:

$$
\begin{aligned}
\Sigma' = \{\ & 0_1, 0_1^f, 1_1, 1_1^f, \lambda_1, \lambda_1^f, 0_2, 0_2^f, 1_2, 1_2^f, \lambda_2, \lambda_2^f, 0_3, 0_3^f, 1_3, 1_3^f, \lambda_3, \lambda_3^f,\\
& 0_1^+, 0_1^-, 1_1^+, 1_1^-, \lambda_1^+, \lambda_1^-, 0_2^+, 0_2^-, 1_2^+, 1_2^-, \lambda_2^+, \lambda_2^-, 0_3^+, 0_3^-, 1_3^+, 1_3^-, \lambda_3^+, \lambda_3^-,\\
& Q, Q_1, Q_2, Q_3, I_1^Q, I_2^Q, I_3^Q, S_1, S_2, S_3, S_4, S_5, S_6,\\
& I_1^{1012Q}, I_2^{1012Q}, I_3^{1012Q}, I_4^{1012Q}, I_1^{1114Q}, I_2^{1114Q}, I_3^{1114Q}, I_4^{1114Q},\\
& I_1^{1\lambda 16}, I_2^{1\lambda 16}, I_3^{1\lambda 16}, I_4^{1\lambda 16},\\
& I_1^{2Q\,302}, I_2^{2Q\,302}, I_3^{2Q\,302}, I_4^{2Q\,302}, I_1^{3Q\,103}, I_2^{3Q\,103}, I_3^{3Q\,103}, I_4^{3Q\,103},\\
& I_1^{4Q\,512}, I_2^{4Q\,512}, I_3^{4Q\,512}, I_4^{4Q\,512}, I_1^{5Q\,113}, I_2^{5Q\,113}, I_3^{5Q\,113}, I_4^{5Q\,113},\\
& w_1, w_2, w_{1012Q}, w_{1114Q}, w_{1\lambda 16}, w_{2Q\,302}, w_{3Q\,103}, w_{4Q\,512}, w_{5Q\,113}\}
\end{aligned}
$$

$$
\begin{aligned}
e' = {} & \lambda_1(0_1 \mid 1_1)^*(0_1^+ \mid 0_1^- \mid 1_1^+ \mid 1_1^- \mid \varepsilon) \mid \lambda_1^+ \mid \lambda_1^- \mid 0_1^f \mid 1_1^f \mid \lambda_1^f\\
& \mid \lambda_2(0_2 \mid 1_2)^*(0_2^+ \mid 0_2^- \mid 1_2^+ \mid 1_2^- \mid \varepsilon) \mid \lambda_2^+ \mid \lambda_2^- \mid 0_2^f \mid 1_2^f \mid \lambda_2^f\\
& \mid \lambda_3(0_3 \mid 1_3)^*(0_3^+ \mid 0_3^- \mid 1_3^+ \mid 1_3^- \mid \varepsilon) \mid \lambda_3^+ \mid \lambda_3^- \mid 0_3^f \mid 1_3^f \mid \lambda_3^f\\
& \mid Q \mid Q_1 \mid Q_2 \mid Q_3 \mid I_1^Q \mid I_2^Q \mid I_3^Q \mid S_1 \mid S_2 \mid S_3 \mid S_4 \mid S_5 \mid S_6\\
& \mid I_1^{1012Q} \mid I_2^{1012Q} \mid I_3^{1012Q} \mid I_4^{1012Q} \mid I_1^{1114Q} \mid I_2^{1114Q} \mid I_3^{1114Q} \mid I_4^{1114Q}\\
& \mid I_1^{1\lambda 16} \mid I_2^{1\lambda 16} \mid I_3^{1\lambda 16} \mid I_4^{1\lambda 16}\\
& \mid I_1^{2Q\,302} \mid I_2^{2Q\,302} \mid I_3^{2Q\,302} \mid I_4^{2Q\,302} \mid I_1^{3Q\,103} \mid I_2^{3Q\,103} \mid I_3^{3Q\,103} \mid I_4^{3Q\,103}\\
& \mid I_1^{4Q\,512} \mid I_2^{4Q\,512} \mid I_3^{4Q\,512} \mid I_4^{4Q\,512} \mid I_1^{5Q\,113} \mid I_2^{5Q\,113} \mid I_3^{5Q\,113} \mid I_4^{5Q\,113}\\
& \mid w_1 \mid w_2 \mid w_{1012Q} \mid w_{1114Q} \mid w_{1\lambda 16} \mid w_{2Q\,302} \mid w_{3Q\,103} \mid w_{4Q\,512} \mid w_{5Q\,113}
\end{aligned}
$$

We give the reaction schemata in Ψ' in multiple groups based on their intended function. The reaction schemata that implement pushing and popping onto the stack are, for each stack $i \in \{1, 2, 3\}$ and symbol $x \in \{0, 1\}$, where λ indicates the bottom of the stack:

$$*_1 \rightleftharpoons *_1 x_i^- \tag{1}$$
$$*_1 x_i^- + x_i^f \rightleftharpoons *_1 x_i^+ \tag{2}$$
$$*_1 x_i^+ \rightleftharpoons *_1 x_i + Q_i \tag{3}$$
$$\lambda_i^- + \lambda_i^f \rightleftharpoons \lambda_i^+ \tag{4}$$
$$\lambda_i^+ \rightleftharpoons \lambda_i + Q_i \tag{5}$$

For each stack i, interchangeability of Q is implemented by:

$$Q_i \rightleftharpoons I_i^Q \tag{6}$$
$$I_i^Q \rightleftharpoons Q \tag{7}$$

The irreversible stack machine transitions as shown in Figure 1 of Qian et al. [42] are incorrect according to CRN bisimulation, as discussed in our previous work: releasing the output species before the first irreversible step allows the reaction to reverse itself, producing a small probability of formally incorrect pathways [27]. As one would expect, this would be incorrect according to PRN bisimulation as well. Qian and Winfree have come up with a corrected version of the DSD mechanism (unpublished), and we give the reaction enumeration of the corrected version below. The stack machine transitions of the form $S_i + A \to S_j + B$, where A and B are either free stack symbols x_k^f or Q (which correspond to the seven classes of I^{iAjB} monomers), are implemented by:

$$S_i \rightleftharpoons I_1^{iAjB} \tag{8}$$
$$I_1^{iAjB} + A \rightleftharpoons I_2^{iAjB} \tag{9}$$
$$I_2^{iAjB} \to I_3^{iAjB} + S_j + w_1 \tag{10}$$
$$I_3^{iAjB} \rightleftharpoons I_4^{iAjB} + B \tag{11}$$
$$I_3^{iAjB} \to w_{iAjB} + w_2 \tag{12}$$

The formal PRN that describes the stack machine is given in Figure 4(d) of Qian et al. [42]. Adapted to our notation, the formal PRN is (Σ, e, Ψ) as follows:

$$\Sigma = \{\, 0_1, 0_1^f, 1_1, 1_1^f, \lambda_1, \lambda_1^f, \lambda_1^-, 0_2, 0_2^f, 1_2, 1_2^f, \lambda_2, \lambda_2^f, \lambda_2^-, 0_3, 0_3^f, 1_3, 1_3^f, \lambda_3, \lambda_3^f, \lambda_3^-,$$
$$Q, Q_1, Q_2, Q_3, S_1, S_2, S_3, S_4, S_5, S_6\}$$

$$e = \lambda_1(0_1 \mid 1_1)^* \mid \lambda_2(0_2 \mid 1_2)^* \mid \lambda_3(0_3 \mid 1_3)^*$$
$$\mid 0_1^f \mid 1_1^f \mid \lambda_1^f \mid \lambda_1^- \mid 0_2^f \mid 1_2^f \mid \lambda_2^f \mid \lambda_2^- \mid 0_3^f \mid 1_3^f \mid \lambda_3^f \mid \lambda_3^-$$
$$\mid Q \mid Q_1 \mid Q_2 \mid Q_3 \mid S_1 \mid S_2 \mid S_3 \mid S_4 \mid S_5 \mid S_6$$

With Ψ containing the reaction schemata:

$$S_1 + 0_1^f \to S_2 + Q$$
$$S_1 + 1_1^f \to S_4 + Q$$
$$S_1 + \lambda_1^f \to S_6 + \lambda_1^f$$
$$S_2 + Q \to S_3 + 0_2^f$$
$$S_3 + Q \to S_1 + 0_3^f$$
$$S_4 + Q \to S_5 + 1_2^f$$
$$S_5 + Q \to S_1 + 1_3^f$$

$$Q \rightleftharpoons Q_i \mid i \in \{1, 2, 3\}$$

$$*_1 + x_i^f \rightleftharpoons *_1 x_i + Q_i \mid x \in \{0, 1\}, i \in \{1, 2, 3\}$$
$$\lambda_i^- + \lambda_i^f \rightleftharpoons \lambda_i + Q_i \mid i \in \{1, 2, 3\}$$

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

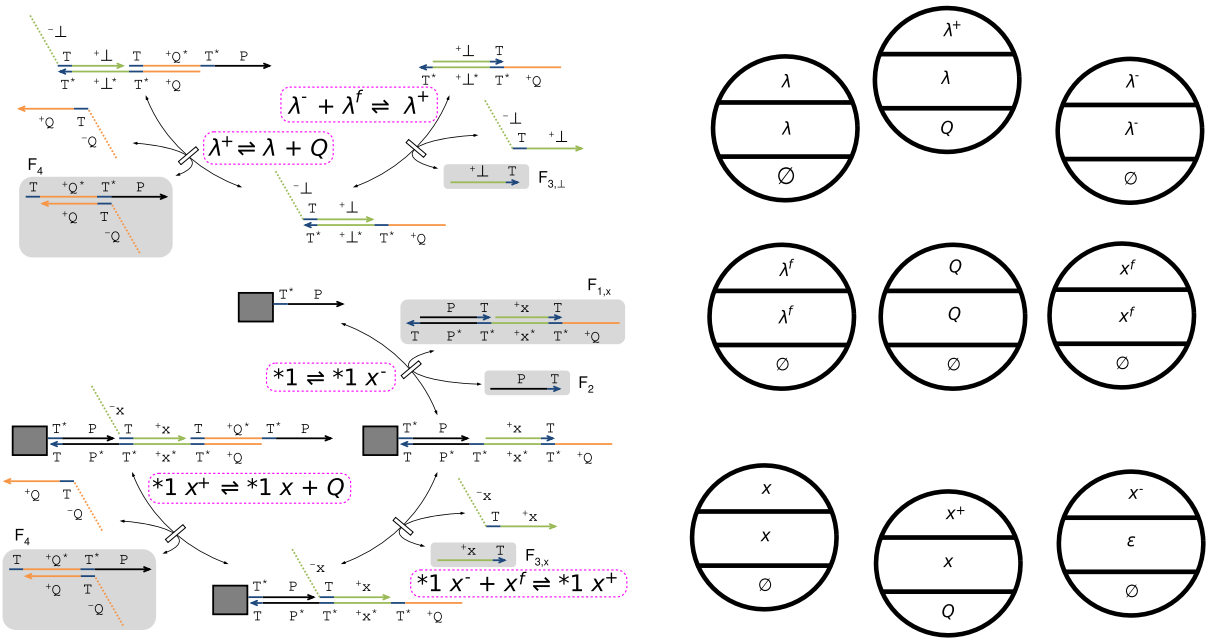


Fig. 6. Enumeration of reaction schemata and interpretation of the monomers in Qian et al.'s DNA stack machine [42] as enumerated in Fig. 5.

Theorem 3.1. *There exists a polymer interpretation (π, μ), from the implementation PRN as described above to the formal PRN as described above, that is a PRN bisimulation.*

The proof of Theorem 3.1, including constructing π and μ, is the remainder of this section. While the proof is given as though for this specific stack machine, it is in fact general and should apply to any instance of the DNA stack machine [42].

Given the formal and implementation PRNs, since we are proving that the implementation is correct, the next step is to construct an interpretation. For the stack machine, as with most engineered implementation systems, the rationale behind the construction suggests a natural interpretation which, if the implementation is correct at all, will be a valid PRN bisimulation. When we give this interpretation, recall the notation $m(x) = (A; B)$ as a shorthand for $\pi(x) = A$, $\mu(x) = B$ where x is an implementation monomer, A a string of formal monomers, and B a multiset of formal species. Here the natural interpretation, shown in Fig. 6, is as follows:

1. A monomer x that appears in both the formal and implementation PRNs, such as $x = S_3$ or $x = 0_2$, has $m(x) = (x; \emptyset)$. Note that this covers all formal monomers.
2. For monomers involved in pushing and popping from the stack, for each stack $i \in \{1, 2, 3\}$, $m(s_i^-) = (\varepsilon; \emptyset)$ for $s \in \{0, 1\}$ and $m(s_i^+) = (s_i; Q)$ for $s \in \{0, 1, \lambda\}$. (The case $m(\lambda_i^-) = (\lambda_i^-; \emptyset)$ is covered as λ_i^- is a formal monomer.)
3. For intermediates I_k^{iAjB} in the stack machine transitions, implementing $S_i + A \to S_j + B$ for $A, B \in \{x_1^f, x_2^f, x_3^f, Q\}$, we have $\pi(I_k^{iAjB}) = \varepsilon$ while $\mu(I_1^{iAjB}) = S_i$, $\mu(I_2^{iAjB}) = S_i + A$, $\mu(I_3^{iAjB}) = B$, and $\mu(I_4^{iAjB}) = \emptyset$. Similarly for the interchange of Q, $m(I_i^Q) = (\varepsilon; Q)$. Each $m(w_{\dots}) = (\varepsilon; \emptyset)$.

Given an interpretation, we check the compatibility condition or the consistency condition to see if CRN bisimulation is even definable. In this case, the stronger, compatibility condition holds: the π-interpretation of any valid implementation species is a valid formal species or ε. Start with e', which describes all valid implementation species: 3 regular expressions describing stack polymers, and a number of species that exist as monomers (i.e., length-1 polymers). The stack polymer expressions are of the form $\lambda_i(0_i \mid 1_i)^*(0_i^+ \mid 0_i^- \mid 1_i^+ \mid 1_i^- \mid \varepsilon)$; since $\pi(0_i^+) = \pi(0_i) = 0_i$, $\pi(1_i^+) = \pi(1_i) = 1_i$, $\pi(0_i^-) = \pi(1_i^-) = \varepsilon$, and $\pi(\lambda_i) = \lambda_i$, any implementation polymer matching this expression will have its π-interpretation match the $\lambda_i(0_i \mid 1_i)^*$ subexpression of e, and thus be a valid formal polymer. The monomers are I-type species whose π-interpretation is ε; formal species Q, Q_i, S_i, x_i^f, and λ_i^- whose π-interpretations are themselves and which appear in e as valid formal length-1 polymers; and the λ_i^+ species whose π-interpretation λ_i matches the $\lambda_i(0_i \mid 1_i)^*$ subexpression of ρ. This covers all cases in e', proving that the compatibility condition holds.

Given the compatibility condition, the induced interpretation m (see Definition 2.9) is in fact a CRN interpretation, and is a PRN bisimulation if and only if it is a CRN bisimulation (Definition 2.13). Thus the last thing we need to do is show that m satisfies the atomic, delimiting, and permissive conditions (Definition 2.4.II). Again, for a PRN treated as an infinite CRN, an algorithmic way of doing this is generally infeasible; in fact we will show in the next section that checking the permissive

condition in the general case for linear PRNs is undecidable. To check an engineered system, one would typically rely on the intent of the designers being formalizable into a proof of correctness. The stack machine was designed in subsystems, each of which correctly implements one formal reaction or formal reaction schema, which when combined form a correct implementation of the formal PRN. The modularity condition from Theorem 2.2 covers exactly this case: we will only need to prove each module correct, and the correctness of the whole system will follow. Thus, the next step is to divide the enumerated formal and implementation CRNs into modules.

To specify each module we must specify a set of formal species, formal reactions, implementation species, implementation reactions, and sets of "common" formal and implementation species, each subsets of their respective sets of species. The modularity condition expects each module to have an interpretation; since we already have m defined, each module's interpretation is just m restricted to its implementation species. For each stack i, we have one module consisting of: all formal species matching $\lambda_i(0_i \mid 1_i)^* \mid \lambda_i^- \mid \lambda_i^f \mid 0_i^f \mid 1_i^f \mid Q_i$, the formal reaction $\lambda_i^- + \lambda_i^f \rightleftharpoons \lambda_i + Q_i$, and all formal reactions enumerated from the two reaction schemata $*_1 + x_i^f \rightleftharpoons *_1 x_i + Q_i \mid x \in \{0, 1\}$ (recall that a reversible reaction schema is a shorthand for two irreversible schemata); all implementation species matching $\lambda_i(0_i \mid 1_i)^*(0_i^+ \mid 0_i^- \mid 1_i^+ \mid 1_i^- \mid \varepsilon) \mid \lambda_i^+ \mid \lambda_i^- \mid 0_i^f \mid 1_i^f \mid \lambda_i^f \mid Q_i$, and all implementation reactions enumerated from reaction schemata of type (1) through (5) for stack i; and the common formal species being all formal species, while the common implementation species are all those with the same name as a formal species. For each stack i, we have a *separate* module containing formal species Q and Q_i and reactions $Q \rightleftharpoons Q_i$; implementation species Q, Q_i, and I_i^Q and reactions of type (6) and (7) for stack i; and $\{Q, Q_i\}$ is again both the set of common formal species and of common implementation species. For each formal reaction of the form $S_i + A \to S_j + B$, we have a module consisting of those formal species and that formal reaction; implementation species S_i, S_j, A, B, and all I_k^{iAjB} species, and reactions of type (8) through (11) for this formal reaction; and again, all formal species are common and all implementation species with the same name as a formal species are common. In this three-stack, six-state, seven-transition stack machine, this gives 13 modules, shown in Fig. 7; we prove below that each of those 13 modules satisfies the atomic, delimiting, permissive, and modularity conditions.

Recall the polymer atomic and polymer delimiting conditions from Theorem 2.3. The argument that the whole system satisfies the atomic condition starts similarly to the polymer atomic condition: each formal monomer x has an implementation monomer with the same name and with $m(x) = (x; \emptyset)$. Then observing that e is a subexpression of e', any formal species $w \in \mathcal{S}$ (i.e., string matching e) will also match e', thus $w \in \mathcal{S}'$ and will have $m(w) = \{\!|w|\!\}$. Because each module, for each formal species it contains, also contains the implementation species with the same name, each module satisfies the atomic condition.

The whole system satisfies the polymer delimiting condition, which we prove by going through the types of implementation reaction schemata. The reader can verify that schemata of types (1) and (3), and reactions of types (5), (7), (8), (9), and (11) are all trivial (for example, type (3) is syntactically interpreted as $*_1 x_i + Q_i \rightleftharpoons *_1 x_i + Q_i$); schemata of type (2) are syntactically interpreted as $*_1 + x_i^f \rightleftharpoons *_1 x_i + Q_i$; reactions of type (4) as $\lambda_i^- + \lambda_i^f \rightleftharpoons \lambda_i + Q_i$; type (6) as $Q_i \rightleftharpoons Q$; and type (10) as the appropriate $S_i + A \to S_j + B$. All of those nontrivial syntactically interpreted reactions or schemata appear in the formal PRN, so the polymer delimiting condition is satisfied, which proves that the whole system satisfies the delimiting condition. Again, each module, for each nontrivial implementation reaction it contains, also contains the corresponding formal reaction, so each module satisfies the delimiting condition.

The permissive condition is where the division into modules matters. To prove the permissive condition we will have to check each formal reaction within each module; since no formal reaction appears in multiple modules, modularity does not increase how much we have to check, and since the size of each implementation module is smaller than the whole implementation CRN, we have less to check per reaction, thus less overall. As discussed in our previous work on CRN bisimulation, we prove the permissive condition by showing that for each formal reaction, for each *minimal* implementation state whose interpretation contains all the formal reactants, that reaction can be implemented; since if every minimal implementation state can do something, then every implementation state can do the same thing [27]. This is illustrated in Fig. 8.

To treat the simple cases first, consider the formal reactions that are not schemata (i.e. have no wildcards). A formal reaction of the form $S_i + A \to S_j + B$ appears only in its own module, in which the minimal implementation states containing $S_i + A$ in their interpretation are $\{\!|S_i + A|\!\}$, $\{\!|I_1^{iAjB} + A|\!\}$, and $\{\!|I_2^{iAjB}|\!\}$. These states implemement $S_i + A \to S_j + B$ by, respectively, forward reactions of type (8) then (9) then (10); (9) then (10); or just (10); (8) and (9) are trivial reactions followed by (10) which is interpreted as $S_i + A \to S_j + B$. As an edge case, if $A = B$ (as is the case for $S_1 + \lambda_1^f \to S_6 + \lambda_1^f$), then any of the above three states with A replaced by I_3^{iAjB} is also a minimal state. Such a state implements $S_i + A \to S_j + B$ by the forward reaction of type (11) followed by the appropriate sequence mentioned above. Similarly, the formal reactions $Q \rightleftharpoons Q_i$ each appear only in their own modules, in which the minimal states for $Q \to Q_i$ are $\{\!|Q|\!\}$ and $\{\!|I_i^Q|\!\}$, and the only minimal state for $Q_i \to Q$ is $\{\!|Q_i|\!\}$. Those three states implement the appropriate formal reaction respectively by forward (6) followed by forward (7); just forward (7); and just reverse (7).

The remaining formal reactions all derive from the stack modules: the $\lambda_i^- + \lambda_i^f \rightleftharpoons \lambda_i + Q_i$ reactions and the reactions enumerated from the $*_1 + x_i^f \rightleftharpoons *_1 x_i + Q_i$ schemata. Each of the three stack modules thus contains infinitely many reactions:

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

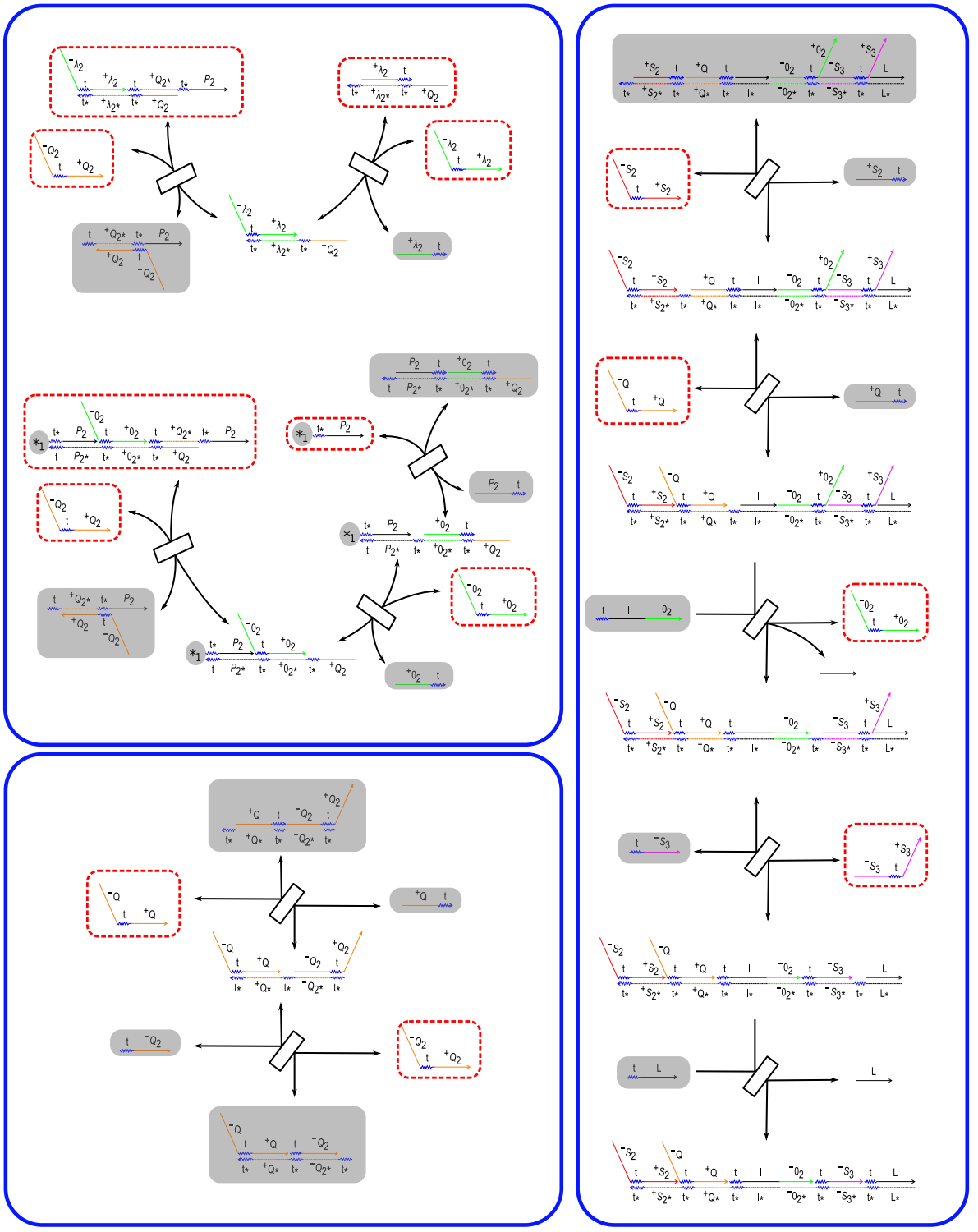


Fig. 7. Examples of the three types of modules. Top left: the stack module for stack 2. Bottom left: the Q exchange module for stack 2. Right: the stack machine transition module for formal reaction $S_2 + Q \rightarrow S_3 + 0_2^f$. Common species are outlined in red dashed lines. DNA complexes covered in gray boxes are fuel complexes, which are not explicitly included in the implementation PRN.

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

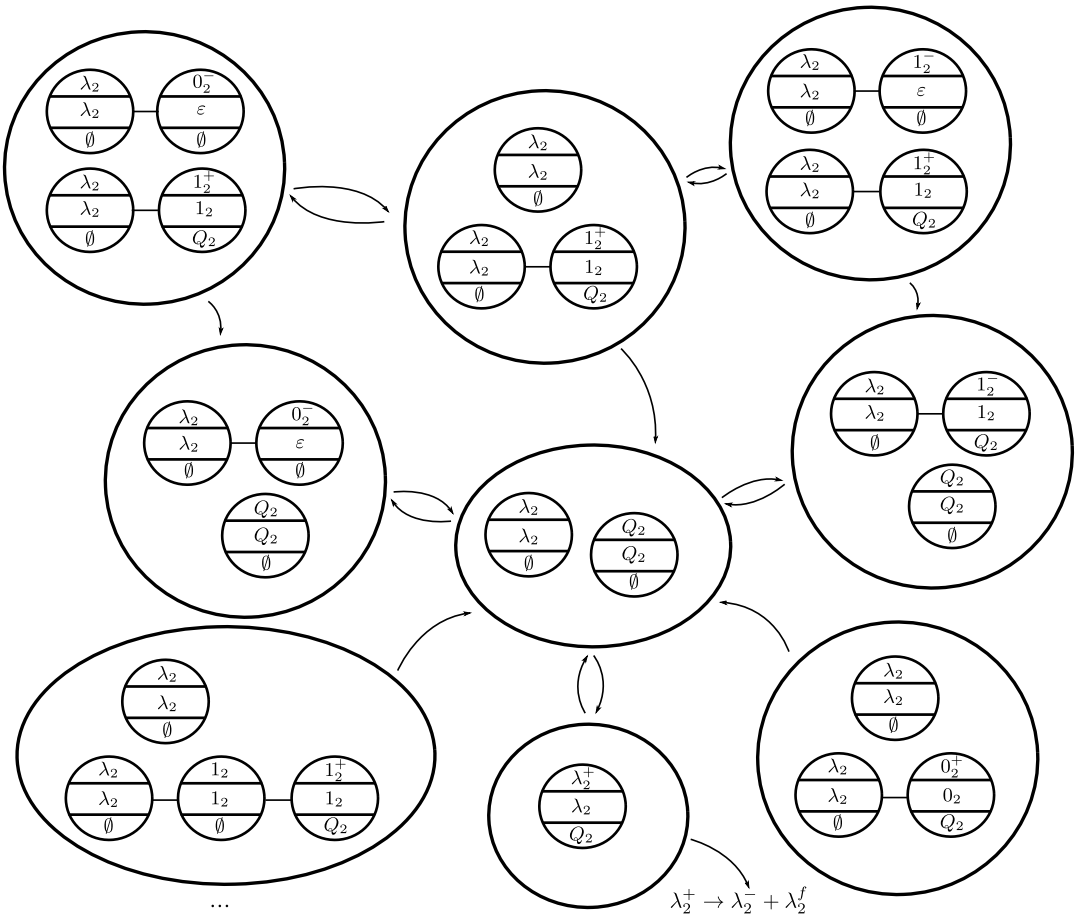


Fig. 8. The "minimal states argument" from our previous work on CRN bisimulation [27] is often the most effective way to prove the permissive condition. Here we show some of the minimal implementation states (within the stack 2 module) in which the formal reaction $\lambda_2 + Q_2 \to \lambda_2^- + \lambda_2^f$ should be able to occur. Arrows between states represent trivial implementation reactions; the arrow with no target represents the implementation reaction $\lambda_2^+ \to \lambda_2^- + \lambda_2^f$, which is interpreted as the desired formal reaction. As in the finite CRN bisimulation case, reversible reactions from a minimal state to a non-minimal state may be shown as irreversible arrows between minimal states, e.g. $*_1 1_2^+ \rightleftharpoons *_1 1_2 + Q_2$ taking $\lambda_2 + \lambda_2 1_2 1_2^+$ to (a non-minimal state containing) $\lambda_2 + Q_2$. Unlike the finite CRN case, here we have infinitely many minimal states (for example, every state of the form $\lambda_2 + \lambda_2 w 1_2^+$, $w \in \{0_2, 1_2\}^*$), so the permissive condition cannot be verified by simply checking for paths in this graph; however, the argument given in the text based on this graph can prove it.

in stack module $i \in \{1, 2, 3\}$, for each $w \in \{0, 1\}^*$ and $x \in \{0, 1\}$, where w_i is w made up of 0_i's and 1_i's, $*_1 + x_i^f \rightleftharpoons *_1 x_i + Q_i$ enumerates a pair of reactions $\lambda_i w_i + x_i^f \rightleftharpoons \lambda_i w_i x_i + Q_i$.

There is only one minimal implementation state for $\lambda_i^- + \lambda_i^f \to \lambda_i + Q_i$, $\{\!|\lambda_i^- + \lambda_i^f|\!\}$, which implements the formal reaction by a forward reaction of type (4). $\lambda_i w_i + x_i^f \to \lambda_i w_i x_i + Q_i$ has four minimal states, namely x_i^f plus any one of $\lambda_i w_i$, $\lambda_i w_i 0_i^-$, $\lambda_i w_i 1_i^-$, $\lambda_i w_i' y_i^+$ if $w = w'y$ for $y \in \{0, 1\}$, or λ_i^+ if $w = \varepsilon$. These states implement the formal reaction as follows: $\lambda_i w_i' y_i^+$ becomes $\lambda_i w_i + Q_i$ by a reaction of type (3); λ_i^+ becomes $\lambda_i + Q_i$ by a reaction of type (5); $\lambda_i w_i (1-x)_i^-$ becomes $\lambda_i w_i$ by a reverse reaction of type (1); $\lambda_i w_i$ becomes $\lambda_i w_i x_i^-$ by a forward reaction of type (1), all of the so-far-mentioned reactions being trivial; and $\lambda_i w_i x_i^- + x_i^f$ implements the formal reaction by a reaction of type (2). The reverse reactions are slightly more complex, because the Q_i can be provided by any implementation species in the module whose interpretation *contains* Q_i, namely Q_i itself, λ_i^+, or any $\lambda_i u_i y_i^+$ for $u \in \{0, 1\}^*$, $y \in \{0, 1\}$. The minimal states for $\lambda_i + Q_i \to \lambda_i^- + \lambda_i^f$ are then either λ_i^+ by itself, or one of λ_i, $\lambda_i 0_i^-$, or $\lambda_i 1_i^-$ plus one of any species providing Q_i (other than λ_i^+, in which case the state would not be minimal). Similarly, the minimal states for $\lambda_i w_i x_i + Q_i \to \lambda_i w_i + x_i^f$ are either $\lambda_i w_i x_i^+$ by itself, or one of $\lambda_i w_i x_i$, $\lambda_i w_i x_i 0_i^-$, or $\lambda_i w_i x_i 1_i^-$ plus one of any non-$\lambda_i w_i x_i^+$ species providing Q_i. These states implement the formal reaction as follows: any species providing Q_i releases the implementation Q_i by a reaction of type (3) or (5) as appropriate; any 0_i^- or 1_i^- "falls off" by a reverse reaction of type (1); free Q_i joins λ_i by a reverse reaction of type (5) or $\lambda_i w_i x_i$ by a reverse reaction of type (3); and finally the formal reaction is implemented by a reverse reaction of type (2) or (4) as appropriate.

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

This covers all modules, and proves that within each module, the permissive condition is satisfied; but does not prove that the permissive condition is satisfied for the whole system.

From our previous work, the modularity theorem proves that, if each module satisfies the permissive condition and the modularity condition, then the whole system satisfies the permissive condition (and the modularity condition) [27]. So the last step is to prove that each module satisfies the modularity condition: each implementation species can "decompose", via trivial reactions, into one multiset of common implementation species and another multiset of implementation species whose interpretation contains no common formal species. For common implementation species, and for implementation species containing no common formal species, this decomposition is already done. The non-common implementation species are the I_k^{iAjB} intermediates, I_i^Q intermediates, λ_i^+, λ_i^-, λ_i^f and x_i^f monomers, and stack polymers matching the expression $\lambda_i(0_i \mid 1_i)^*(0_i^+ \mid 0_i^- \mid 1_i^+ \mid 1_i^- \mid \varepsilon)$, but of those only I_k^{iAjB} for $k \neq 4$, I_i^Q, λ_i^+, and species matching $\lambda_i(0_i \mid 1_i)^*(0_i^+ \mid 1_i^+)$ contain common formal species. Each of those species decomposes as follows: I_1^{iAjB} to S_i via reverse reaction (8); I_2^{iAjB} to $A + S_i$ via reverse reactions (9) and (8); I_3^{iAjB} to $I_4^{iAjB} + B$ via reaction (11); I_i^Q to Q via reaction (7); λ_i^+ to $\lambda_i + Q_i$ via reaction (5); and a species of the form $\lambda_i w_i x_i^+$ to $\lambda_i w_i x_i + Q_i$ via a reaction enumerated from reaction schema (3). This satisfies the modularity condition, meaning that the permissive condition will be satisfied when the initial implementation state combines species from different modules. This completes the proof that the given (π, μ) is a PRN bisimulation from the DNA stack machine to its formal description, which implies that the two systems will have the same set of trajectories from any initial state. Since it is intuitive that the formal system, when started in the appropriate initial states, performs a rate-independent simulation of an abstract stack machine, so does the DNA system.

4. Hardness results

Having defined a concept of correctness of an implementation of a polymer network, we would like to be able to algorithmically check, given two polymer networks and an interpretation, whether that interpretation is a bisimulation. Knowing that polymer networks are capable of Turing-universal computation, we might suspect that to be impossible. A next best thing would be if bisimulation or non-bisimulation was recursively enumerable: either that any correct interpretation would have a proof of correctness, or that any incorrect interpretation would have a proof of incorrectness. Unfortunately, neither one is the case. We show that verifying our notion of bisimulation for PRNs is equivalent to the uniform halting problem, which given a Turing machine, asks if every possible configuration of the Turing machine will eventually lead to a halting configuration [24]. This problem is in the class Π_2^0, the complement of the second level of the arithmetic hierarchy, which is the class of all languages $L = \{x \mid \forall_y \exists_z \phi(x, y, z)\}$, where ϕ is a decidable predicate. Since each level of the arithmetic hierarchy strictly contains the previous levels, a Π_2^0-complete problem cannot be recursively enumerable, nor can its complement [30]. Since the uniform halting problem is Π_2^0-complete [24], so is PRN bisimulation. It is also interesting to note that the atomic condition, which is trivial to check for finite CRNs, becomes PSPACE-complete for Polymer Reaction Networks, proven by reduction from the problems of checking whether a regular expression describes the language of all strings, or whether two regular expressions describe the same language or languages one of which contains the other [26,50,39].

Lemma 4.1. *Given a formal species schema (Σ, e), implementation species schema (Σ', e'), and interpretation (π, μ), the problem of checking the atomic condition and that of checking the compatibility condition are both PSPACE-complete. If one or both are required to be local, or equivalently, given as a compatibility relation (Σ, ρ) and/or (Σ', ρ'), then checking the atomic condition remains PSPACE-complete. In contrast, checking the compatibility condition can be done in polynomial time if the formal schema is known to be local, while it remains PSPACE-complete if only the implementation schema is known to be local.*

Proof. To show that a decision problem is PSPACE-complete, we must show that (a) it is in PSPACE, i.e. that it can be decided by a Turing machine that uses polynomial space, and (b) that it is complete, e.g. by exhibiting a polynomial-time reduction from a known PSPACE-complete problem to this one. The problems of, given a pair of regular expressions (e_1, e_2) over Σ, deciding whether the language of e_1 is contained in that of e_2, deciding whether the languages are identical, and deciding whether $L(e_1) = \Sigma^*$, are all PSPACE-complete [26,50,39] (and there are simple reductions between them).

Recall that the CRN atomic condition requires that for every formal (polymer) species, there is an implementation species interpreted as exactly one copy of that formal species. The compatibility condition states that the π-interpretation of a valid implementation polymer cannot result in an invalid formal polymer: $x_1 \ldots x_n \in L(e')$ implies $\pi(x_1) \ldots \pi(x_n) \in L(e[+])$.

To check the atomic condition in polynomial space, we choose $e_1 = e$ to describe the set of formal polymer species allowed in the formal PRN, while e_2 will describe the set of interpretations of polymers in the implementation that correspond to exactly one formal polymer. This is laborious but straightforward: e_2 is an expression for π-interpretations of strings matching e' that polymerize as a single formal polymer and carry nothing, union with the (finite) set of $A \in \Sigma^*$ produced as the only formal polymer carried in the μ-interpretation of an implementation polymer that polymerizes as nothing. The CRN atomic condition holds if $L(e_1) \subset L(e_2)$. (The compatibility condition implies that $L(e_2) \subset L(e_1)$, in which case these two sets of formal polymer species need to be identical. However, for PRN bisimulation up to reachability, the consistency condition may allow for implementation species to exist that do not correspond to formal polymers, so long as they will not be reached. In that case the atomic condition still requires $L(e_1) \subset L(e_2)$.) To construct the first part of

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

e_2, we start with e' and restrict it to use only monomers $x \in \Sigma'$ with $\mu(x) = \emptyset$, replace each valid monomer with its π-interpretation, and intersect with $+^*\Sigma^*+^*$. This can be done in polynomial time. To construct the second part of e_2, we look for strings $A \in \Sigma^*$ such that some x_0 has $\pi(x_0) = \varepsilon$ and $\mu(x_0) = \{\!|A|\!\}$ and there exist u, v with $ux_0v \in L(e')$ where all monomers x in uv have $m(x) = (\varepsilon, \emptyset)$, i.e. they are "empty monomers". Given an NFA for e', which can be obtained in polynomial time from e', this is simple: identify all states of the NFA reachable from the start using only empty monomers, identify all states that can reach an accepting state using only empty monomers, and identify all desired x_0 that transition between them. Then with e_1 and e_2 constructed, we use the known PSPACE algorithm for testing the containment of languages described by regular expressions.

To show completeness, we are now given arbitrary e_1 and e_2 for the regular expression containment problem. To reduce this problem to checking the atomic condition for PRNs, we are free to choose a formal PRN with arbitrary e, along with an implementation PRN containing the same monomers but a different regular expression e', and an interpretation where every monomer polymerizes as itself and carries nothing. It follows that we can set $e = e_1$ and $e' = e_2$ such that the atomic condition holds iff $L(e_1) \subset L(e_2)$. Thus, checking the atomic condition is PSPACE-complete.

Similarly, to check the compatibility condition, $e_2 = e[+]$ and e_1 is the expression of π-interpretations of strings matching e', this time regardless of their μ-interpretations, and we use the known PSPACE algorithm that decides containment for regular expressions. Again, proving completeness uses $\Sigma' = \Sigma$ and $m(x) = (x; \emptyset)$, which allows reduction from regular expression containment since our interpretation does not use "+". Given arbitrary e_1 and e_2, $L(e_1) \subset L(e_2)$ if and only if the compatibility condition is satisfied for a formal PRN (Σ, e_2) and implementation PRN (Σ, e_1).

If one or both species schemata are given as local regular expressions or compatibility relations, then we recall Lemma 2.2, that given any regular expression there is a compatibility relation on an implementation monomer set and a π-interpretation under which they allow the same set of strings. Here we use that deciding whether a regular expression e matches all strings over Σ is also PSPACE-complete [26,50,39]. For the atomic condition, consider a formal species schema (Σ, ρ) with $\rho = \Sigma_{\vdash} \times \Sigma_{\dashv}$ (allowing all strings) and implementation schema (Σ', ρ') and π-interpretation implementing e according to Lemma 2.2. In such a case, the atomic condition is true if and only if e matches all of Σ^*. This proves PSPACE-completeness of the atomic condition when both formal and implementation regular expressions are local, which *a fortiori* also applies when only one is local.

We have already established the PSPACE-completeness of checking the compatibility condition when both formal and implementation schema are allowed to be nonlocal; here we will cover the other three cases. It turns out that whether the implementation schema is local does not affect the complexity. When the formal schema is allowed to be nonlocal, consider an unrestricted implementation PRN (Σ, Σ^*) and formal PRN (Σ, e) with an arbitrary regular expression e, where $m(x) = (x; \emptyset)$. Σ^* is of course local and the implementation is correct (with respect to compatibility) if and only if $L(e) = \Sigma^*$, thus proving that checking the compatibility condition remains PSPACE-complete even when the implementation schema must be local. However, when the formal schema is required to be local—or equivalently, given by a compatibility relation—the result changes. Given a formal schema (Σ, ρ) and implementation schema (Σ', e') where e' may be local or not, we show how to test the compatibility condition in polynomial time. We first check that for every implementation monomer x, there are no violations of ρ within $\pi(x)$ itself; this requires checking boundary conditions (with $\vdash$ and $\dashv$) only if and where $\pi(x)$ contains a $+$ symbol, because we don't yet know whether x can begin or end a polymer according to e'. If this test passes, we must further test that when the π-interpretations of monomers in an implementation polymer are strung together, there are no violations of ρ at the boundaries. A complication is that since a monomer may polymerize as nothing, the π-interpretation of a monomer may abut the π-interpretation of another monomer that is distant in the implementation polymer. We check this with a pseudo-compatibility relation ρ'_ε which represents implementation monomers that can be connected to each other by zero or more monomers that polymerize as nothing, which can then be compared to the formal ρ. Define ρ'_ε such that $x\rho'_\varepsilon y$ if any $uxvyw \in \mathcal{S}(\Sigma', e')$, $u, v, w \in (\Sigma')^*$, with $\pi(v) = \varepsilon$; the case $x = \vdash$ (resp. $y = \dashv$) corresponds to $vyw \in \mathcal{S}(\Sigma', e')$ (resp. $uxv \in \mathcal{S}(\Sigma', e')$) with the same restrictions on u, v, w. This can be computed in polynomial time with reachability questions on the nondeterministic finite automaton associated with e'. It is in general not true that $\mathcal{S}(\Sigma', e') = \mathcal{S}(\Sigma', \rho'_\varepsilon)$, but where $\pi(x)_1$ and $\pi(x)_{-1}$ are the first and last characters, respectively, of $\pi(x)$, it is true that the compatibility condition is true if and only if for all $x \in \Sigma' \cup \{\vdash\}$ and $y \in \Sigma' \cup \{\dashv\}$ with $\pi(x) \neq \varepsilon \neq \pi(y)$, $x\rho'_\varepsilon y \Rightarrow \pi(x)_{-1}\rho\pi(y)_1$, with the convention $\pi(\vdash)_{-1} = \vdash$ and $\pi(\dashv)_1 = \dashv$, and where $+\rho y \iff \vdash\rho y$, $x\rho+ \iff x\rho\dashv$. Thus when the formal schema is local, the compatibility condition can be computed in polynomial time. □

Theorem 4.1. *The problem of, given a formal PRN (Σ, e, Ψ), implementation PRN (Σ', e', Ψ'), and interpretation (π, μ), deciding whether that interpretation is a bisimulation is Π^0_2-complete.*

Proof. Weak bisimulation is the statement that *for all* pairs of related states and steps in one of the two states *there exists* a corresponding sequence of steps in the other state, which is naturally a Π^0_2 statement. (In PRN bisimulation this description applies to both the delimiting and permissive conditions, while the atomic condition is decidable in PSPACE by Lemma 4.1.) To prove completeness, we reduce from the uniform halting problem: given a Turing machine, is true that from any combination of state and tape contents, the Turing machine halts? Since PRNs can simulate Turing machines, we show that the condition that, for all states of a PRN, a given reaction can happen is equivalent to the condition that, for all configurations of a Turing machine, the Turing machine will halt. In the case of PRN bisimulation, the above condition corresponds to the

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

permissive condition, in an implementation PRN where the delimiting condition is true. Since the uniform halting problem is Π_2^0-complete [24], so is bisimulation.

Given a Turing machine M with alphabet $\{0, 1\}$ (blank squares treated as 0), with set of states Q including a start state q_0 and halt state q_H, we construct a pair of PRNs and an interpretation which is a bisimulation if and only if M halts from every instantaneous description. The formal PRN is $(\{Q, H\}, Q \mid H, \{Q \to H\})$. The implementation PRN is the simulation of M generalized from Example 2.2, simulating M using state monomers q_i and tape squares 0^l and 1^l to the left of the state, 0^r and 1^r to the right, and between one and six reaction schemata for each transition in M. $\mu(x) = \emptyset$ for all x, $\pi(0^l) = \pi(0^r) = \pi(1^l) = \pi(1^r) = \varepsilon$, $\pi(q_i) = Q$ for each non-halting state q_i, and $\pi(q_H) = H$.

Given $e' = (0^l \mid 1^l)^*(q_i)(0^r \mid 1^r)^*$ from the generalized Example 2.2, the valid implementation polymers are exactly the valid instantaneous descriptions of M, and the only reactions that can happen are simulations of steps of M. Any valid implementation species has only one state q_i, and thus interprets to either Q or H, both of which are valid formal species, which also satisfies the atomic condition. Any implementation reaction is a transition of M, so the corresponding formal step is either trivial, if the transition is not to q_H, or $Q \to H$ if it is, satisfying the delimiting condition. In any formal state with a Q, and any implementation state interpreted as that formal state, there is at least one polymer representing a non-halting instantaneous description of M, and the statement that all such states can eventually do $Q \to H$ (the permissive condition) is equivalent to the statement that all instantaneous descriptions eventually halt. □

While the general case is undecidable, it would be valuable to identify restricted (yet useful) classes where PRN bisimulation can be efficiently determined algorithmically.

5. Single-locus networks

Given a class of interesting Polymer Reaction Networks, we would naturally want to find a physical implementation of some or all of those networks. So far, theoretical steps taken towards implementing polymer reactions with DNA nanotechnology include the stack machine implementations by Qian et al. [42] and by Lakin and Phillips [33], the Turing machine implementation by Yahiro and Hagiya [55], the register machine by Tai and Condon [51], and the surface CRN implementations by Qian and Winfree [43]. To illustrate one challenge in implementation, recall the string copying and equality/reverse detection PRNs from Fig. 3. For example, the one-step string copying PRN uses reaction schemata of the form

$$P + A{*_1} \to P + A *_1 + A *_1 .$$

While this schema describes the copying of an arbitrarily long string starting with A and catalyzed by P, physical systems (biological, engineered, or otherwise) tend not to copy arbitrarily long strings in one step. The local model string-copying PRN in Fig. 3 transcribes a string of length n in $O(n)$ steps, each of which affects only a constant number of monomers (specifically, at most 3). In general, physical systems will—on the most realistic level—be modeled as such local and bounded reactions, by which we mean reactions that only "read" and "write" a finite number of monomers and/or connections between monomers.

If we try to model the local mechanism of DNA polymerase as an implementation of $P + A{*_1} \to P + A *_1 + A{*_1}$, an immediate problem is that the structure is no longer linear, but branched. This problem is somewhat related to an issue with naively enumerating a PRN from a DNA strand displacement system: in the stack machine, for example, treating a single strand as a monomer will fail when some strands have enough domains to bind to three other strands at once. In that case, since the "third branch" never exceeded a fixed size, a clever choice of DNA complexes to be treated as implementation monomers allowed us to model the system as a linear PRN, but the same is not true for DNA polymerase. A DNA polymerase "implementation" network could be modeled in Cardelli's Biochemical Ground Form [13], or in the branching PRN extended model we discuss in Section 6, but not in the linear Polymer Reaction Network model. Even if we use a model with branching polymers, the implementation will not be correct according to bisimulation: in the formal network, the second copy of the arbitrarily long polymer $A{*_1}$ is produced in one step, which is impossible in an implementation network made up of only local and bounded reactions. (The network could be correct according to CRN bisimulation on the induced infinite CRNs, where for each polymer w the branched structure being built up from an initial $A{*_1} = w$ is interpreted as $w + P$ until the final dissociation step, at which point each copy of w is interpreted as w. However, PRN bisimulation would require each individual monomer to have an interpretation, preventing this workaround.)

The key obstacle here is the (so far informal) concept of "local and bounded", and the difficulty of implementing formal reaction schemata that are not "local and bounded" using only implementation schemata that are. (Or at least, the difficulty of doing so in a way bisimulation can recognize and verify.) For the moment, therefore, we will turn to implementation of reaction schemata that *are* local and bounded, with a suitable definition. We define a concept of a *single-locus reaction schema*, which we feel captures the informal concept of "local and bounded" (see Fig. 9). We will show that these single-locus reaction schemata can be implemented up to bisimulation by a set of four polymer primitives, three of which have candidate DNA implementations from the stack machine [42] or surface CRNs [43]. We show that a class of infinite CRNs, which is intuitively the class of single-locus PRNs plus compatibility relation-based computational power, is closed under bisimulation and any member of that class can be implemented by the given primitives, suggesting that the concept of single-locus schemata is a natural class to discuss.

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

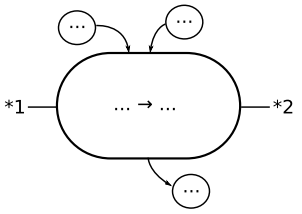


Fig. 9. Single-locus reaction schemata are, intuitively, schemata whose reactions occur entirely within one region whose size is not affected by wildcards. This region may be in the middle of one polymer (if $*_1$ and $*_2$ are on different edges of the same polymer), at the joining of two polymers (if $*_1$ and $*_2$ are on separate polymers), or at either end of one polymer (if either $*_1$ and/or $*_2$ does not exist). The schema may consume, produce, and/or be catalyzed by any number of additional, finite polymers, since such a reaction can still be thought of as taking place within a finite region.

Definition 5.1. A reaction schema is *single-locus* if:

(i) Any wildcard that appears at all, appears exactly once in the reactants and exactly once in the products.
(ii) Wildcards appear only at the beginning or at the end of a polymer, and each wildcard that appears, appears at the same place (beginning or end) in the products as in the reactants.
(iii) All wildcards that appear at the beginning of a polymer must have the same numerical subscript, and likewise for all wildcards that appear at the end of a polymer.

For the purpose of conditions (ii) and (iii), a wildcard that is the entire polymer can be counted as at the beginning or at the end, and the reaction schema is single-locus if it satisfies the conditions for at least one of those two choices. For example, $*_1 \rightleftharpoons *_1 E$, $*_1 \rightleftharpoons E*_1$, and $*_1 + *_2 \rightleftharpoons *_1 I *_2$ are all single-locus.

A PRN (Σ, e, Ψ) is single-locus if e is local and each reaction schema in Ψ is single-locus. An augmented PRN (Σ, e, Ψ) is *augmented single-locus* if each reaction schema in Ψ, ignoring its regular expression restrictions, is single-locus; in the augmented case e is not required to be local.

Intuitively, a reaction schema is single-locus if it only "reads from" (is conditional on) and "writes to" (changes) one region of finite size, i.e. containing no wildcards. $*_1 AB*_2 \to *_1 CDE*_2$ is the ideal example of this; it "reads" AB and "writes" CDE, a region of size at most 3, while leaving $*_1$ and $*_2$ unchanged. Similarly, in $*_1 A *_2 + B \to *_1 C *_2$, the "region" includes both the region A on the polymer $*_1 A *_2$ and the free monomer B, which since the monomer B has no wildcards is still a finite size. A reaction schema $*_1 + A*_1 \to *_1 + B*_1$ would *not* be single-locus, since to check that both polymers have the same sequence substituted for the wildcard $*_1$ requires reading that sequence, and a wildcard's sequence is not bounded by a finite size. Similarly, $*_1 A *_2 B *_3 \to *_1 C *_2 D *_3$ requires reading both the A and B and writing both C and D, which thanks to the intervening $*_2$ cannot be included in one region of finite size. (These examples correspond to violations of conditions (i) and (ii), respectively.) A schema such as $*_1 A *_2 + *_3 B *_4 \to *_1 C *_2 + *_3 D *_4$ is not single-locus by the above definition because it violates condition (iii), even though the A and B could be viewed as a finite region to read from and write to. To do so, however, we would have to view the A-B region as a single region on a branched polymer, and for the same reason, implementing this reaction schema (up to PRN bisimulation) with "physically possible" (i.e., single-locus) reactions of *linear* polymers is impossible. We will, however, return to this topic in Section 6. An augmented single-locus PRN is not exactly local, and reaction schemata may read (but not change) unbounded regions; however, it turns out that augmented single-locus PRNs are a natural class of PRNs closed under PRN bisimulation.

Theorem 5.1. *For any formal single-locus PRN (Σ, e, Ψ), there is a PRN $(\Sigma', (\Sigma')^*, \Psi')$ and bisimulation up to reachability interpretation (π, μ) such that all reaction schemata in Ψ' are of one of the following four forms:*

$$*_1 AB*_2 \to *_1 CD*_2 \qquad \text{(Context-sensitive Replacement)}$$

$$*_1 A *_2 + B \rightleftharpoons *_1 C *_2 \qquad \text{(Monomer-dependent Replacement)}$$

$$*_1 \rightleftharpoons *_1 E \quad \text{or} \quad *_1 \rightleftharpoons F *_1 \qquad \text{(Reversible Addition)}$$

$$*_1 + *_2 \rightleftharpoons *_1 I *_2 \qquad \text{(Reversible End-joining)}$$

Proof. This proof is centered around a way to implement any reaction schema of the form

$$*_1 x_1 \dots x_n *_2 + r_1 + \cdots + r_k \to *_1 y_1 \dots y_m *_2 + p_1 + \cdots + p_l,$$

where x_i and y_i are monomers, $r_i = r_{i,1} \dots r_{i,n_i}$ and similarly p_i are strings of monomers. First we show how to convert any single-locus PRN into that form. Reactants or products of the form $*_1 w_1 + w_2 *_2$ (here w_i are strings of monomers) can be replaced by $*_1 w_1 I w_2 *_2$ together with the reaction schema $*_1 + *_2 \rightleftharpoons *_1 I *_2$, where $m(I) = (+; \emptyset)$. Reactants or products without $*_1$ (resp. $*_2$) can replace $w*_2$ with $*_1 F_L w *_2$ with the reaction schema $*_1 \rightleftharpoons F_L *_1$ (resp. replace $*_1 w$

with $*_1 w F_R *_2$ and add the reaction schema $*_1 \rightleftharpoons *_1 F_R$), where $m(F_L) = m(F_R) = (\varepsilon; \emptyset)$. This argument implicitly makes use of the transitivity property of CRN bisimulation [27], which applies equally well to infinite CRNs and thus to PRNs. For example, it is simple to confirm that (for any reasonable Σ, e, Σ', e') $\{*_1 AIB *_2 \rightarrow *_1 C *_2, *_1 + *_2 \rightleftharpoons *_1 I *_2\}$ is a correct (up to PRN bisimulation) implementation of $\{*_1 A + B *_2 \rightarrow *_1 C *_2\}$, so by transitivity, any correct implementation of the former PRN will be a correct implementation of the latter. We also assume that $ux_1 \dots x_n v \in L(e) \Rightarrow uy_1 \dots y_m v \in L(e)$; since e is local, this is easily checkable in terms of the corresponding compatibility relation ρ: we require that $\{z \mid (z, x_1) \in \rho\} = \{z \mid (z, y_1) \in \rho\}$, and similarly $\{z \mid (x_n, z) \in \rho\} = \{z \mid (y_m, z) \in \rho\}$. If this is not the case, we can replace this schema with multiple schemata of the form $*_1 x_0 x_1 \dots x_n x_{-1} *_2 + \dots \rightarrow *_1 x_0 y_1 \dots y_m x_{-1} *_2 + \dots$ for every combination x_0 and x_{-1} allowed by ρ (and consider each such schema separately), each of which trivially satisfies the condition. Such a replacement will again be a correct implementation up to bisimulation of the original single schema, and again transitivity applies. Given an implementation of each reaction schema in a formal PRN, combining the implementation reaction schemata will produce a correct implementation of the formal PRN; this relies on the modularity property of CRN bisimulation [27], and in fact the implementation we give will satisfy the condition for modularity to hold.

Given a formal reaction schema of the above form, we can implement it as follows: use $*_1 AB *_2 \rightleftharpoons *_1 CE *_2$ and/or $*_1 A *_2 + B \rightleftharpoons *_1 C *_2$ trivial reactions to combine all reactants into two monomers on one polymer; use a $*_1 AB *_2 \rightarrow *_1 CD *_2$ reaction to convert those two into two monomers representing the products; finally use the reverse of the first process to separate those into the intended products. In the implementation CRN we have a monomer a for each formal monomer A, and a monomer for each prefix w of x, y, or any r_i or p_i. (If a string w is a prefix of multiple such strings, they will use the same w monomer.) We have an implementation monomer E with $m(E) = (\varepsilon, \emptyset)$, and reaction schemata $*_1 \rightleftharpoons *_1 E$ and $*_1 w E *_2 \rightleftharpoons *_1 E w *_2$ for every prefix monomer w (including formal monomers as prefixes of length 1). Where w is a prefix of any of the above and wA is the next prefix, we have a reaction schema $*_1 wA *_2 \rightleftharpoons *_1 E(wA) *_2$, where wA on the left means the two monomers w and A while (wA) on the right means the one monomer for the prefix wA; with these schemata, we can collect, reversibly, any prefix of any individual reactant or product into one monomer. We have monomers r^i for $1 \le i \le k$ and p^i for $1 \le i \le l$; where r_i (resp. p_i) refers to the "prefix" monomer that is the entire string of the formal species r_i (resp. p_i) and $r^0 = p^0 = E$, we have reaction schemata $*_1 r^{i-1} *_2 + r_i \rightleftharpoons *_1 r^i *_2$ (resp. $*_1 p^{i-1} *_2 + p_i \rightleftharpoons *_1 p^i *_2$); with these schemata, we can collect, reversibly, all reactants or products into two monomers on one polymer. Finally, we have a reaction schema to convert the two monomers representing the reactants into the two monomers representing the products: where x (resp. y) is the prefix monomer for the entire string $x_1 \dots x_n$ (resp. $y_1 \dots y_m$), we have the reaction schema $*_1 r^k x *_2 \rightarrow *_1 p^l y *_2$. As an edge case, if $n = 0$ the reactants of that last reaction are $*_1 r^k *_2$, if $k = 0$ the reactants are $*_1 x *_2$, and if $n = k = 0$ the reactants are $*_1 E *_2$; the products are treated similarly if m and/or $l = 0$. We let $e' = (\Sigma')^*$; anything can bind to anything else, but we rely on the reaction schemata to keep the polymers formally valid and the consistency condition to ensure that they do. We show that this is a correct implementation, according to modular PRN bisimulation up to reachability, of the given formal reaction schema.

To show that this implementation is correct, we construct an interpretation; show that it satisfies the consistency condition; then show that it satisfies the atomic, delimiting, permissive, and modularity conditions. The interpretation is intuitive: where w is an implementation monomer that is a string of formal monomers, $m(w) = (w; \emptyset)$, $m(r^i) = (\varepsilon; \sum_{j=1}^{i} r_j)$, $m(p^i) = (\varepsilon; \sum_{j=1}^{i} p_j)$, and $m(E) = (\varepsilon, \emptyset)$. The consistency condition then follows from the assumption that $\{x_0 \mid (x_0, x_1) \in \rho\} = \{x_0 \mid (x_0, y_1) \in \rho\}$ and $\{x_{-1} \mid (x_n, x_{-1}) \in \rho\} = \{x_{-1} \mid (y_m, x_{-1}) \in \rho\}$: the only reaction schema that changes the π-interpretation of any polymer is the intended formal schema, $*_1 r^k x *_2 \rightarrow *_1 p^l y *_2$, which replaces an x_1 after $*_1$ and x_n before $*_2$ with y_1 after $*_1$ and y_m before $*_2$. (The $*_1 r^{i-1} *_2 + r_i \rightleftharpoons *_1 r^i *_2$ and similar p^i schemata create and destroy π-interpretations, but those r_i and p_i are by assumption valid formal species.) This allows m as a CRN interpretation to be defined.

The atomic condition follows from the polymer atomic condition, which is satisfied by the formal monomers as implementation monomers. The delimiting condition follows from the polymer delimiting condition: it is simple to confirm that all reaction schemata are syntactically interpreted as trivial except $*_1 r^k x *_2 \rightarrow *_1 p^l y *_2$, which is syntactically interpreted as the single formal reaction schema. To prove the permissive condition, it is simpler to prove the modularity condition first, with respect to all formal species as common formal species and all polymers made of only formal species as common implementation species. We thus show that any implementation species can be decomposed, via trivial reactions, into common implementation species. Given an arbitrary non-common implementation species, decompose it as follows: first, use $*_1 r^i *_2 \rightarrow *_1 r^{i-1} *_2 + r_i$ and $*_1 p^i *_2 \rightarrow *_1 p^{i-1} *_2$ schemata to produce a set of species with only prefix monomers and E monomers. Observe that $*_1 \rightleftharpoons *_1 E$ and $*_1 w E *_2 \rightleftharpoons *_1 E w *_2$ schemata can take any such polymer to any other such polymer with the same sequence of prefix monomers interspersed with any pattern of E's. In particular, for each polymer in the current decomposition, take that polymer to one where each prefix monomer w is to the right of exactly $|w| - 1$ E monomers. From such a state, $*_1 E(wA) *_2 \rightarrow *_1 wA *_2$ schemata will produce polymers with only formal monomers, finishing the decomposition to only common implementation species.

Given that every non-common implementation species can be decomposed via trivial reactions to common implementation species, we need only prove the permissive condition from minimal states consisting of only common species. For each formal reaction, i.e. each choice of w_1 and w_2 to be substituted for $*_1$ and $*_2$, exactly one such minimal state exists: $w_1 x_1 \dots x_n w_2 + r_1 + \dots + r_n$. This minimal state implements the formal reaction by the intuitive path: $*_1 wA *_2 \rightarrow *_1 E(wA) *_2$ reactions to reach $w_1 E^{n-1} x w_2$ (in the edge case where $k > 1$ and $n = 0$ or $n = 1$, use $*_1 \rightarrow *_1 E$ and $*_1 w E *_2 \rightarrow *_1 E w *_2$ to reach $w_1 E x w_2$); $*_1 r^{i-1} *_2 + r_i \rightarrow *_1 r^i *_2$ reactions with the initial $r^0 = E$ on the E directly to

the left of x, reaching $w_1E^{n-2}r^kxw_2$ (in the edge case $k=0$ ignore this step; in the edge case $n<2$ the result will be $w_1r^kxw_2$); then the reaction $w_1E^{n-2}r^kxw_2 \to w_1E^{n-2}p^lyw_2$ is enumerated from $*_1r^kx*_2 \to *_1p^ly*_2$ and is interpreted as $w_1x_1\dots x_nw_2 + r_1 + \dots + r_k \to w_1y_1\dots y_mw_2 + p_1 + \dots + p_l$, satisfying the permissive condition. Any minimal state within this module implements that formal reaction by first decomposing all non-common implementation species then following the above path; any minimal state from outside this module satisfies the permissive condition by the modularity theorem; so this completes the proof that this interpretation is a PRN bisimulation up to reachability. □

Initially we expected the class of single-locus PRNs would be closed under PRN bisimulation, but quickly found a counterexample: a formal PRN with reaction schemata $A *_1 X \to A *_1 Y$ and $B *_1 X \to B *_1 Z$ is not single-locus, but can be implemented by single-locus reaction schemata $*_1x_A \to *_1y$ and $*_1x_B \to *_1z$ where $\pi(x_A) = \pi(x_B) = X$, $\pi(y) = Y$, $\pi(z) = Z$, if the implementation compatibility relation guarantees that x_A can only appear in a polymer whose interpretation begins with A, and x_B only in a polymer whose interpretation begins with B. Intuitively, a single-locus implementation schema has "computational power" equal to the computational power of its formal syntactic interpretation (in the sense of the polymer delimiting condition, Theorem 2.3) plus that of the regular expression restriction. (Given Lemma 2.2, this extra power would still be present had we defined PRNs using compatibility relations instead of regular expressions.) It is also important to note that in Definition 2.13 we defined PRN bisimulation as roughly a (π, μ) polymer interpretation whose induced CRN interpretation m is well-defined and is a CRN bisimulation, which cares about the set of implementation and formal reactions but not about the set of reaction schemata from which they were enumerated. Thus our statement about closed classes takes the form, "given a (possibly infinite) formal CRN and a single-locus implementation PRN with polymer interpretation that is a CRN bisimulation, the set of formal reactions is equal to the set of reactions enumerated from some set of augmented single-locus formal reaction schemata".

The proof given below requires m to use the wildcard-based interpretation (Definition 2.10) because it depends on, for each implementation schema, "syntactically interpreting" it to create a formal schema. Conveniently, the definition of single-locus schema implies that the multisets of wildcards in reactants and in products of a schema are equal, so the wildcard-based interpretation is defined for any reaction that can only be enumerated in one way from one schema. The process of "syntactically interpreting" a schema inherently interprets wildcards as wildcards, discarding any members of the μ-interpretation of monomers in wildcards, thus naturally corresponding to the wildcard-based interpretation, which is why we argue that is the "natural" definition of what is a spurious catalyst in a polymer reaction. If an alternate spurious catalyst interpretation is used, the proof and/or the theorem may or may not hold; we suspect a sufficiently complex spurious catalyst interpretation may effectively add "non-single-locus behavior" to a PRN, in which case we would not even want the theorem to hold.

Theorem 5.2. *Let (Σ', e', Ψ') be a single-locus implementation PRN, (Σ, e) a formal species schema, and $\mathcal{R}$ a set of formal reactions such that $(\mathcal{S}(\Sigma, e), \mathcal{R})$ is a formal CRN. Let (π, μ) be a polymer interpretation between the above species schemata that satisfies the consistency condition and whose induced interpretation m, using the wildcard-based interpretation for spurious catalysts in each reaction, is a CRN bisimulation. Then there is some set Ψ_0 of augmented single-locus reaction schemata such that $\mathcal{R}(\Sigma, e, \Psi_0) = \mathcal{R}$. Conversely, given an augmented single-locus PRN (Σ, e, Ψ_0) there is an implementation PRN (Σ', e', Ψ') where all schemata in Ψ' are of the types described in Theorem 5.1 with PRN bisimulation interpretation (π, μ). If e is local, then (Σ', e', Ψ') is single-locus, and further $(\Sigma', (\Sigma')^*, \Psi')$ is also single-locus and the same (π, μ) defined on that PRN is a PRN bisimulation up to reachability.*

Proof. Given (Σ', e', Ψ'), (Σ, e), $\mathcal{R}$, and (π, μ), we produce a set Ψ_0 of augmented single-locus reaction schemata with $\mathcal{R}(\Sigma, e, \Psi_0) = \mathcal{R}$. Recall the concept of "syntactically interpreting" a reaction schema, as used in Theorem 2.3: replace each implementation monomer with its π-interpretation and add its μ-interpretation to the appropriate side of the reaction schema, producing a reaction schema defined in terms of formal monomers. The desired Ψ_0 is the set of syntactic interpretations ψ_i of each reaction schema $\psi_i' \in \Psi'$ (which, given that syntactic interpretations preserve the placement of wildcards, will be single-locus). For each $*_1$ (or $*_2$) in ψ_i', because the schema is single-locus, it appears as either $*_1x', \dots, \dots x'*_1$, $*_1*_2$, or $*_1$ alone. In either case, the set of possible implementation sequences preceding or following some x', or forming the first (or last) part of a polymer, or forming an entire polymer, can be described by a regular expression over Σ'. The regular expression restriction $e_{i,1}$ (or $e_{i,2}$) is obtained from this regular expression by replacing each implementation monomer with its π-interpretation.

It follows from the three conditions of CRN bisimulation that the set of formal reactions $\mathcal{R}$ equals the set of nontrivial interpretations of implementation reactions in $\mathcal{R}(\Sigma', e', \Psi')$. (If π satisfies the consistency condition but not the compatibility condition, then this is true for the set of nontrivial interpretations of implementation reactions whose reactants are interpreted as valid formal species.) Then given any reaction enumerated from some $\psi_i' \in \Psi'$ (with the above condition if π only satisfies the consistency condition), its interpretation will be enumerated from the corresponding ψ_i: whatever values $*_1$ and $*_2$ take in the implementation enumeration, their π-interpretations will be the values of $*_1$ and $*_2$ in the formal enumeration. Those values, by construction, will satisfy the regular expressions $e_{i,1}$ and $e_{i,2}$, and the full (i.e., combining π and μ) interpretation of the monomers in ψ_i' will be the monomers and extra polymers in ψ_i; the compatibility or consistency condition, as appropriate, ensures that the formal interpretations match e so that the reaction is in fact enumerated. (If the μ-interpretation of implementation monomers in $*_1$ or $*_2$ is nonempty, then those monomers will appear as both reactants and products, and its wildcard-based interpretation, which must appear in $\mathcal{R}$ for m to be a CRN bisimulation, is the reaction

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

without those spurious catalysts.) Given any reaction enumerated from some $\psi_i \in \Psi_0$, similarly consider the corresponding $\psi_i' \in \Psi'$. By construction, the regular expression restrictions on ψ_i represent all strings that are π-interpretations of some string of implementation monomers that would be a valid substitution for the appropriate wildcard; those strings for the formal values of $*_1$ and $*_2$ will be the implementation values of $*_1$ and $*_2$. Recalling that the spurious catalysts definition of bisimulation removes nonempty μ-interpretations, the interpretation of the implementation reaction so produced will be the formal reaction in question. Thus the set of reactions enumerated from Ψ_0 with restrictions is the set of interpretations of nontrivial reactions enumerated from Ψ', which since m is a CRN bisimulation is equal to $\mathcal{R}$.

Given a formal augmented single-locus PRN $(\Sigma, e, \{\psi\})$ with one reaction schema, we construct an unaugmented implementation PRN (Σ', e', Ψ') and PRN bisimulation interpretation (π, μ), where every reaction schema in Ψ' is single-locus, and if e is local then so is e'. Given Theorem 5.1 and the transitivity and modularity results, this is sufficient to prove the statement of this theorem. Effectively, we will construct an implementation PRN that uses (non-augmented) single-locus trivial reactions to check that the wildcards satisfy their regular expression restrictions, and if the check passes, implements ψ. The information that the check passed will be stored next to the "single locus" that ψ affects, thus making the implementation single-locus without augmentation.

Say ψ takes the form $*_1 x_1 \dots x_n *_2 + r_1 + \dots + r_k \to *_1 y_1 \dots y_m *_2 + p_1 + \dots + p_l$, where some x_i and/or y_i may be $+$ (if $*_1$ and $*_2$ are on different polymers), and $*_1$ is restricted to match the regular expression e_1 while $*_2$ must match e_2. Let M_1 be an NFA recognizing e_1 and M_2 an NFA recognizing the reverse of e_2. Let Σ' be Σ together with, for each $x \in \Sigma$ and q a state in M_1 or M_2, two additional species x_q and x_q'. Construct e' as the intersection of two regular expressions as follows. First, replacing each x_q and x_q' with x should produce a string that matches e. Second, starting from the leftmost monomer may trace a valid partial computation of M_1 as follows: $k-1$ monomers of the form $(x_i)'_{q_i}$, $0 < i < k$, followed by a monomer $(x_k)_{q_k}$, such that where q_0 is the start state of M_1, $q_{i-1} \xrightarrow{x_i} q_i$ for $1 \le i \le k$; and starting from the rightmost monomer reading right-to-left may trace a valid partial computation of M_2 in the same manner, while between these partial computations only monomers $x \in \Sigma$ appear. Observe that because the partial computation regular expression is local, if e is local then so is e'.

Ψ' will have reversible reaction schemata

$$x *_1 \rightleftharpoons x_q *_1 \text{ for } (\vdash, x) \in \rho,\ q_{0;M_1} \xrightarrow{x} q;$$

and

$$*_1 x_q y *_2 \rightleftharpoons *_1 x_q' y_s *_2 \text{ for } (x, y) \in \rho,\ q \xrightarrow{y} s \in M_1$$

to simulate M_1; similarly

$$*_1 x \rightleftharpoons *_1 x_q \text{ for } (x, \dashv) \in \rho,\ q_{0;M_2} \xrightarrow{x} q$$

and

$$*_1 x y_q *_2 \rightleftharpoons *_1 x_s y_q' *_2 \text{ for } (x, y) \in \rho,\ q \xrightarrow{x} s \in M_2$$

to simulate M_2; and

$$*_1 x_q x_1 \dots x_n y_s *_2 + r_1 + \dots + r_k \to *_1 x_q y_1 \dots y_m y_s *_2 + p_1 + \dots + p_l$$

for any pair of formal monomers (x, y) and any pair (q, s) of accepting states of M_1 and M_2 respectively. (For clarity, in that last reaction schema $x_1 \dots x_n$ and $y_1 \dots y_m$ are the monomers in Σ that appear in ψ, while x_q, y_s, etc. are monomers $x, y \in \Sigma$ subscripted with NFA states.) The polymer interpretation will have

$$\pi(x) = \pi(x_q) = \pi(x_q') = x$$

and

$$\mu(x) = \mu(x_q) = \mu(x_q') = \emptyset$$

for all x, q.

The construction of e' implies that π satisfies the compatibility condition, and $\Sigma \subset \Sigma'$ with strings matching e also matching e' implies the atomic condition. For the delimiting condition, first, any reaction enumerated from one of the reversible reaction schemata will be trivial. For any reaction enumerated from the last schema, which matches the formal schema, to be made of valid implementation species, it must have $*_1 x_q$ be an accepting computation of M_1 and $y_r *_2$ (the reverse of) an accepting computation of M_2, implying that the corresponding formal strings match e_1 and e_2. The same requirement implies that the interpreted string matches e, which means that the interpreted formal reaction is a reaction enumerated from ψ with the restrictions; thus the delimiting condition is satisfied. For the permissive condition, observe that any implementation polymer containing partial computations can reverse itself to a formal polymer, thus proving modularity (with respect to both sets of common species being $\mathcal{S}$). Starting from only formal polymers whose

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

interpretation can implement a reaction enumerated from Ψ, the applicable implementation polymers can use the reversible reaction schemata to simulate M_1 on the beginning and M_2 on the end, at which point the nontrivial schema applies and the formal reaction can be implemented.

If e is not local, then applying Theorem 5.1 with transitivity and modularity to any number of reaction schemata in (Σ, e, Ψ_0) produces a (Σ', e', Ψ') with single-locus schemata in Ψ' (since the classes of schemata in Theorem 5.1 are all single-locus), but the PRN itself is not single-locus since e' is not local. However, if e is local, then so is e', so the same (Σ', e', Ψ') is single-locus. In that case, given the compatibility relation ρ that is equivalent to e, we can remove from the Ψ' any schema where the interpretations of two adjacent monomers are not compatible according to ρ. (For example, the last schema would only be present for (x, x_1), (x_n, y), (x, y_1), and (y_m, y) all $\in \rho$.) Since we are no longer relying on e' to ensure that enumerated reactions respect e, we can define a PRN $(\Sigma', (\Sigma')^*, \Psi')$ and guarantee that (π, μ) defined on that PRN will satisfy the consistency condition, since the only reaction schemata defined are those that respect ρ everywhere outside the wildcards and leave monomers adjacent to wildcards unchanged. For the same reason as shown in Theorem 5.1, this (π, μ) is a PRN bisimulation up to reachability. □

While, as mentioned above, the class of single-locus PRNs are not closed under PRN bisimulation, combining Theorem 5.2 with the transitivity of PRN bisimulation provides us with the desired related result that the class of augmented single-locus PRNs *are* closed under PRN bisimulation.

Corollary 5.3. *Let (Σ', e', Ψ') be an augmented single-locus implementation PRN, (Σ, e, Ψ) a formal PRN, and (π, μ) a PRN bisimulation interpretation. Then $\mathcal{R}(\Sigma, e, \Psi) = \mathcal{R}(\Sigma, e, \Psi_0)$ where (Σ, e, Ψ_0) is an augmented single-locus PRN.*

6. Alternate polymer models and extended models

In defining linear Polymer Reaction Networks and PRN bisimulation, we made various choices of model properties. Alternative choices would have led to different models, some of which would have similar theorems applicable, some of which would have had different results. Here we briefly discuss two of those alternative choices, and what effects they would have had on the above theory.

6.1. Altered wildcards

Recall again the one-step and local string copying and comparison models in Fig. 3. As opposed to the previous section, here we pay attention to what behaviors we can define if we don't care about single-locus restrictions on wildcards. If we want to copy, move, or remove an arbitrary polymer, or compare two polymers, wildcards as defined can do that:

$$*_1 + P \rightarrow *_1 + *_1 + P$$

On the other hand, consider a simplified model of RNA polymerase, written (in not-yet-defined notation) as:

$$*_1\{A, T, C, G\} + P \rightarrow *_1 + P + *_1[A \rightarrow a, T \rightarrow u, C \rightarrow c, G \rightarrow g]$$

Here RNA polymerase acting on a polymer made up of the DNA bases A, T, C, and G produces a copy replacing each DNA base with the corresponding RNA base a, u, c, or g, similar to the result of the string copying local model. String transcription is not significantly stranger than string copying, and it seems reasonable to construct a model that, if it can describe one as a one-step process, can do the same for both. We might similarly want to model effects that have a wildcard and its reverse, such as polymerase reverse-copying a single strand of DNA, or a stack and its reverse meeting and annihilating each other (also shown as a multi-step mechanism in Fig. 3), or possibly other transformations of wildcards.

One way to define such a model is as follows: In a reaction schema, each wildcard $*_i$ must, in exactly one spot in the reactants, be written $*_i\{A_1, \ldots, A_n\}$, for some set of monomers $\{A_j\}$. (As a notational convenience, $*_i\{\Sigma\}$ can be written as just $*_i$.) At any other point in the reactants and/or products where $*_i$ appears, it can appear as $*_i[A_1 \rightarrow B_1, \ldots, A_k \rightarrow B_k]$, and/or be tagged $*_i^{rev}$. Such a schema is enumerated as follows: $*_i\{A_1, \ldots, A_n\}$ is replaced by a string w_i containing only the A_j's, and modified instances of $*_i$ are replaced by w_i reversed and/or with each A_j replaced by B_j, as appropriate. Other tags, with the same syntax as reverse and with corresponding modifications in the semantics, could be defined as necessary. More generally, wildcards in reactants and products could be related by finite state transducers [45]. Schemata that use these features, however, would seldom be single-locus, since these features involve reading and/or writing arbitrarily large strings in the wildcards.

Mostly, the main content of this paper is orthogonal to this aspect of the model. A PRN with this extension is still enumerated into and treated as a (probably infinite) CRN; PRN bisimulation is still defined as previously discussed; the hardness results still apply. Single-locus PRNs are defined (as they should be) to exclude these features, so those results are similarly unaffected. Overall, we did not define PRNs with these features because we did not need these features to discuss the DNA stack machine, but the model should handle these features without too much difficulty.

It is also of interest when physically rotating a polymer in a way that reverses left and right causes the same molecule to be described by a different string of monomers. For example, say we want to represent fully double-stranded DNA as a

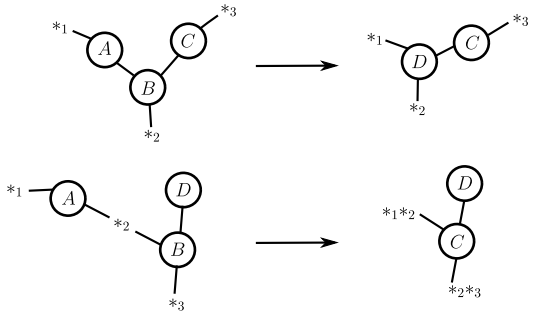
Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

linear polymer, and we let the monomer T represent a T base on the top strand paired with an A on the bottom strand, and similarly for A, C, and G. Then e.g. the strings $TGGC$ and $GCCA$ represent the same physical molecule, and any good model should recognize that. To handle this we could say that in a *rotatable PRN*, there is some function on monomers $x \to x^t$ with $(x^t)^t = x$, which extends to polymers such that $(uv)^t = u^t v^t$, and the species of the enumerated CRN are pairs of equivalent strings $\{w, w^t\}$. Reactions are enumerated from schemata such that for some substitution into the wildcards, whatever strings are produced, the pairs containing those strings are the CRN species involved in the enumerated reaction. This requires intuitive restrictions on the compatibility relation, $(x, y) \in \rho \iff (y^t, x^t) \in \rho$ where $\vdash^t = \dashv$, and on polymer interpretations, $\pi(x^t) = \pi(x)^t$ and $\mu(x^t) = \mu(x)$. The rest of the theory should be compatible without further changes. As an example, if the string-reverse detection local model from Fig. 3 is taken as a rotatable PRN with $0_l^t = 0_r$, $1_l^t = 1_r$, $S^t = S$, and $Y^t = Y$, then it will identify a string over $\{0_l, 1_l\}$ with its reverse over $\{0_r, 1_r\}$, and two copies of the same such string will go through the mechanism that eventually produces Y.

6.2. Branched polymers

Examples of Turing-universal computation used in molecular programming, such as register machines [13,51], stack machines [42,33], and Turing machines themselves [43,55], tend to be linear, so we studied a system of linear PRNs: each monomer can bind to at most two other monomers, and we can write a polymer as a string of monomers. We could instead have allowed each monomer to make either an arbitrary number of bonds, or up to some finite (characteristic of the monomer) number of bonds, either of which would allow us to model much more general systems. Such an approach would present some complications for defining reaction schemata, and present further complications for defining bisimulation, but we believe those complications are all solvable.

The obvious way to extend our definition of linear Polymer Reaction Networks to more general PRNs is effectively a graph-rewriting system with wildcards [44]. In this sense, a (linear) polymer reaction schema such as $*_1 A *_2 + B \to *_1 C *_2$ is already a graph rewriting rule, where all graphs must be lines, and this is a straightforward generalization. Two examples of this would be:



When we defined polymer interpretations for PRN bisimulation, we defined a π-interpretation and a μ-interpretation. With linear polymers, it was easy to say that, given a string of monomers, we interpret them by concatenating their π-interpretations. The equivalent for nonlinear polymers, if a monomer's π-interpretation is anything other than a single monomer, is not obvious. One solution might be to say that each monomer must have a finite set of "faces", i.e. potential bonds, and for each face of an implementation monomer, its π-interpretation specifies a face of a formal monomer in the π-interpretation to correspond to that implementation face. As a special case, we could say that an implementation monomer whose π-interpretation is ε must have at most two faces, and if it has two faces both connected to something, the connections are connected to each other in the interpreted formal polymer. This concept of faces would also solve a similar problem with defining the compatibility relation, and would allow linear PRNs as defined above to be a subcase of branched PRNs, where every monomer has exactly two faces, "left" and "right". The rest of the theory of PRN bisimulation should extend naturally to branched PRNs. To develop this idea into a working theory, chemistry-inspired graph rewriting approaches might be a good place to start [1].

Single-locus PRNs can be defined for branched PRNs, and in fact can be defined in a somewhat more natural way than for linear PRNs. Recall Definition 5.1 of linear single-locus PRNs, in particular condition (iii), that no two distinct wildcards appear at the beginning of a polymer, and similarly for the end. This definition was motivated by that, when imagining a physical implementation of a single-locus reaction schema, such an implementation of a schema that violates condition (iii) would require an intermediate step that is not linear. When the underlying PRN model allows branched polymers, this is not as much of a problem. For branched PRNs, we would define single-locus reaction schemata as follows: (i) any wildcard that appears at all, appears exactly once in the reactants and exactly once in the products, and (ii) all wildcards have at most one bond. (Of the above branched reaction schemata, the first is single-locus and the second is in multiple ways not single-locus.) We suspect a theorem similar to Theorem 5.1 would be provable for branched single-locus PRNs.

The Biochemical Ground Form (BGF), discussed by Cardelli and Zavattaro [13], serves as an example of what a branched polymer model could look like. (The concept of faces, for example, corresponds roughly to association labels in the BGF.) The

BGF, instead of graph rewriting reaction schemata, defines reactions in terms of the actions of different "agents" (monomers), some of which may require coordination with other agents. Implicitly, if a monomer A can take an action a, it can do so regardless of what that monomer is bound to, which in our way of writing means every reaction (schema) has all possible wildcards. In particular, we suspect every BGF system could be written a single-locus branched PRN. The BGF as described has no mechanism for a monomer to coordinate *specifically with another monomer bound to it*, as opposed to another monomer of the specific type that may be on a different polymer; with such a mechanism, we suspect but have not proven that the BGF could implement, up to bisimulation, any single-locus branched PRN.

Extending bisimulation from branched polymers to molecules with arbitrary graph topology, again with reactions implemented as graph rewriting rules [44,1], would also be an interesting research direction. Indeed, certain types of bisimulation for graph grammars have been studied [6].

6.3. Relation to string rewriting models

Linear Polymer Reaction Networks have a number of close connections to string rewriting systems [8], not the least of which being that objects are simply represented as strings. For example, semi-Thue systems, which allow derivations wherein at each step a substring u_i from a given set is replaced by the corresponding v_i, can be modeled exactly by a PRN with the unimolecular reactions $*_1 u_i *_2 \to *_1 v_i *_2$. Semi-Thue systems are already Turing universal [40]. (For example, the copy-tolerant Turing machine in Fig. 3 illustrates this point.) Not only do PRNs allow a wide range of rewriting operations, but they also allow bimolecular reactions and reactions with more than one product, making them in our opinion more appropriate and natural for modeling chemical reaction networks involving polymers (despite that the computational power is not increased). Similarly, branched Polymer Reaction Networks would be related to term rewriting systems [4] as the main representation is a tree; again, the notion of bimolecular reactions and counts of species are novel to the chemistry context.

L-systems [41] are another related string rewriting model from the study of natural computing that has been used effectively to model growth processes ranging from chains of dividing bacteria to multicellular plant development. Like cellular automata, L-systems are one-dimensional and invoke a simultaneous parallel update semantics, but unlike cellular automata, each cell (character of the string) is replaced by either a length-1 string (cell state update) or a longer string (growth) or an empty string (death). While synchronous updates are not natural in the PRN model, unimolecular PRN reaction schemata can directly model asynchronous variants of both cellular automata and L-systems. More generally, our PRN model can be considered an example of a string-based artificial chemistry [19,22].

7. Discussion

Our main claim is that polymer CRN-like systems are a strong candidate for powerful and practical molecular computation; that formal verification is useful for systematic construction of (eventually, large) polymer systems; and that bisimulation is a useful technique in formal verification of polymer systems. To show this, we defined a model of linear Polymer Reaction Networks, and defined PRN bisimulation based on that model. We proved some useful properties of PRN bisimulation; we showed how to use PRN bisimulation to verify an existing system; and we showed an example of how PRN bisimulation can identify good design strategies for implementing a large class of systems. Although we did all of this within the model of linear PRNs, we discussed how PRN bisimulation is likely to be applicable, and our results translatable, to other models of polymer CRN-like systems. Thus, even if this model of linear PRNs is not the optimal model for polymer systems in molecular programming, the concept of PRN bisimulation will likely remain useful.

Our definition of PRN bisimulation interprets each state of the implementation system as a state of the formal system, and checks whether, from any initial state, the possible trajectories of the two systems are equivalent under that interpretation. However, it ignores quantitative aspects of the system such as rate constants, meaning PRN bisimulation says nothing about the kinetics of the system (i.e. how long things take) or the probabilities of the various possible trajectories. It also assumes that the model of the implementation system as a PRN is accurate, and the model we used in this case ignores the "leak reactions" and other side reactions typical of DNA strand displacement systems; with no way to distinguish between likely and unlikely reactions, PRN bisimulation evaluated on a model including leak reactions would say that the implementation is incorrect. This means that *when an implementation is proven correct according to PRN bisimulation, we know that a specific class of its behavior is equivalent to the corresponding behavior of its specification (formal PRN), namely the rate-independent behavior up to (if applicable) whatever model we used to describe the implementation system as an implementation PRN.* In systems such as stack machines and Turing machines, the rate-independent behavior is the only relevant behavior of the abstract system, so PRN bisimulation proves that the implementation has the behavior we want. In systems such as oscillators or dynamic instability, while PRN bisimulation can prove some correspondence between the implementation and the abstract system, it may not be able to say anything about the kinetics that imply the relevant behavior. (Whether an extension of PRN bisimulation can take kinetics into account is, as it is for CRN bisimulation [27], an important open question.) Intuitively we expect that for "systematic implementations" such as the stack machine or the various CRN translation schemes, if the scheme has no qualitative (i.e., detectable by CRN/PRN bisimulation) errors then its kinetics are "close enough" to and/or can be tuned to match those of the abstract system. Experimental implementations such as the CRN oscillator by Srinivas et al. [49] suggest this is the case, and the experiments of Chen et al. [17] demonstrate an experimentally working

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

CRN even when CRN bisimulation identifies a potential error (that, presumably, averages out). In our experience, polymer systems (compared to well-mixed CRNs) are more likely to depend on rate-independent computation and not care about kinetics; for example, well-mixed CRNs require kinetics to approximate the behavior of "A happens, then B happens", while polymer systems can use geometric separation to achieve the same thing with less probability of error. (This is more often true for polymer systems that simulate classic models of computation than for those found in biology.) Consequently, while PRN bisimulation cannot prove correct every relevant aspect of an implementation PRN in general, it is a useful tool to verify the important aspects of many useful polymer systems.

The simplest thing to do with PRN bisimulation is to, given one formal PRN and one putative implementation, verify by hand that the implementation matches the formal PRN. We demonstrated an example of this with the DNA stack machine from Qian et al. [42]. (Lakin and Phillips [33] also provide verification of their improved DNA stack machine implementation, but unlike our symbolic approach that guarantees correctness for all possible inputs, their application of probabilistic model checking relies on explicit enumeration of the state space for specific inputs. This is likely to catch systematic errors in the implementation, but not guaranteed to.) We suspect that bisimulation can be used in more powerful ways, such as automated verification of systems too large to verify by hand, or as a basis for formal proofs that certain classes of systems will or will not be correct implementations of other classes, or as an intuition to guide designers of molecular devices in their search for correct implementations.

As we would expect for a model equivalent in power to Turing machines, whether two systems are PRN bisimulation equivalent is undecidable in general, but this does not rule out any form of computer-aided verification. Exactly what form such verification could take, we don't know, but we have two possibilities to suggest. The main problem that produces the undecidability result stems from the permissive condition, that for every formal reaction in any implementation state whose interpretation can do that reaction, the implementation state can implement the formal reaction after some sequence of zero or more trivial reactions. The problem is that there is no upper bound on the number of trivial reactions; the undecidability result uses a formal reaction that can be implemented only if a Turing machine computation made of trivial reactions halts. Systems intended to be built in practice typically use a small, and in particular bounded, number of trivial reactions per formal reaction. Based on this, the first suggestion is that some bound on the number of trivial reactions may give a definition of PRN bisimulation that is decidable or even tractable. Exactly what type of bound is best, and whether this idea covers all the physical implementations we care about, is unknown. Similarly, systems intended to be built in practice typically have a designer who knows how the system is intended to work, and can provide a "proof" that the permissive condition is satisfied, as we did for the DNA stack machine above. The hardness result shows that not every correct implementation will have a finite proof at all, let alone one that can be checked in reasonable time, but it may be that a large enough class of "reasonable" implementations does. How exactly such a proof should be specified, and what class of systems can be proven correct this way, is unknown.

That formal verification methods such as PRN bisimulation can be used to guide design is a speculation of ours. We showed a concrete example of this idea with the proof that any "physically realistic" (single-locus) PRN can be implemented by five reaction schema "primitives". This sort of result will likely be helpful for designing complex polymer systems, where whatever complex behavior the designer needs can be implemented in a known way with simple primitives, which themselves can be implemented in some known way yet to be discovered. We further hope that, with a formal definition in mind of what makes a correct implementation, someone designing physical implementations would have a better idea of what systems to design.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank Chris Thachuk, Damien Woods, Dave Doty, Seung Woo Shin, and Lulu Qian for helpful discussions. RFJ and EW were supported by NSF grants 1317694, 1213127, and 0832824. RFJ was also supported by Caltech's Summer Undergraduate Research Fellowship and an NSF graduate fellowship.

References

[1] J.L. Andersen, C. Flamm, D. Merkle, P.F. Stadler, A software package for chemically inspired graph transformation, in: International Conference on Graph Transformation, Springer, 2016, pp. 73–88.
[2] D. Angluin, J. Aspnes, D. Eisenstat, Stably computable predicates are semilinear, in: Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, ACM, 2006, pp. 292–299.
[3] D. Angluin, J. Aspnes, D. Eisenstat, A simple population protocol for fast robust approximate majority, Distrib. Comput. 21 (2008) 87–102.
[4] F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, 1999.
[5] S. Badelt, C. Grun, K.V. Sarma, B. Wolfe, S.W. Shin, E. Winfree, A domain-level DNA strand displacement reaction enumerator allowing arbitrary non-pseudoknotted secondary structures, J. R. Soc. Interface 17 (2020) 20190866.
[6] P. Baldan, A. Corradini, U. Montanari, Bisimulation equivalences for graph grammars, in: Formal and Natural Computing, Springer, 2002, pp. 158–187.
[7] C.H. Bennett, The thermodynamics of computation––a review, Int. J. Theor. Phys. 21 (12) (1982) 905–940.

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007

[8] R.V. Book, F. Otto, String-Rewriting Systems, Springer, 1993.
[9] L. Cardelli, Two-domain DNA strand displacement, Math. Struct. Comput. Sci. 23 (2013) 247–271.
[10] L. Cardelli, M. Tribastone, M. Tschaikowski, A. Vandin, Forward and backward bisimulations for chemical reaction networks, in: 26th International Conference on Concurrency Theory (CONCUR 2015), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.
[11] L. Cardelli, M. Tribastone, M. Tschaikowski, A. Vandin, Syntactic Markovian bisimulation for chemical reaction networks, in: Models, Algorithms, Logics and Tools, Springer, 2017, pp. 466–483.
[12] L. Cardelli, M. Tribastone, M. Tschaikowski, A. Vandin, Comparing chemical reaction networks: a categorical and algorithmic perspective, Theor. Comput. Sci. 765 (2019) 47–66.
[13] L. Cardelli, G. Zavattaro, On the computational power of biochemistry, in: Algebraic Biology, Springer, 2008, pp. 65–80.
[14] G. Chatterjee, N. Dalchau, R.A. Muscat, A. Phillips, G. Seelig, A spatially localized architecture for fast and modular DNA computing, Nat. Nanotechnol. 12 (2017) 920.
[15] H.-L. Chen, D. Doty, D. Soloveichik, Deterministic function computation with chemical reaction networks, Nat. Comput. 13 (2014) 517–534.
[16] H.-L. Chen, D. Doty, D. Soloveichik, Rate-independent computation in continuous chemical reaction networks, in: Proceedings of the 5th Conference on Innovations in Theoretical Computer Science (ITCS), 2014, pp. 313–326.
[17] Y.-J. Chen, N. Dalchau, N. Srinivas, A. Phillips, L. Cardelli, D. Soloveichik, G. Seelig, Programmable chemical controllers made from DNA, Nat. Nanotechnol. 8 (2013) 755–762.
[18] A. Desai, T.J. Mitchison, Microtubule polymerization dynamics, Annu. Rev. Cell Dev. Biol. 13 (1997) 83–117.
[19] P. Dittrich, J. Ziegler, W. Banzhaf, Artificial chemistries—a review, Artif. Life 7 (2001) 225–275.
[20] A. Dobrinevski, E. Frey, Extinction in neutrally stable stochastic Lotka-Volterra models, Phys. Rev. E 85 (2012) 051903.
[21] D. Doty, M. Hajiaghayi, Leaderless deterministic chemical reaction networks, Nat. Comput. 14 (2015) 213–223.
[22] J.-L. Giavitto, G. Malcolm, O. Michel, Rewriting systems and the modelling of biological systems, Comp. Funct. Genomics 5 (2004) 95–99.
[23] D.T. Gillespie, Exact stochastic simulation of coupled chemical reactions, J. Phys. Chem. 81 (1977) 2340–2361.
[24] G.T. Herman, Strong computability and variants of the uniform halting problem, Math. Log. Q. 17 (1971) 115–131.
[25] A. Hjelmfelt, E.D. Weinberger, J. Ross, Chemical implementation of neural networks and Turing machines, Proc. Natl. Acad. Sci. 88 (1991) 10983–10987.
[26] H.B. Hunt III, D.J. Rosenkrantz, T.G. Szymanski, On the equivalence, containment, and covering problems for the regular and context-free languages, J. Comput. Syst. Sci. 12 (1976) 222–268.
[27] R.F. Johnson, Q. Dong, E. Winfree, Verifying chemical reaction network implementations: a bisimulation approach, Theor. Comput. Sci. 765 (2019) 3–46.
[28] N.D. Jones, L.H. Landweber, Y.E. Lien, Complexity of some problems in Petri nets, Theor. Comput. Sci. 4 (1977) 277–299.
[29] R.M. Karp, R.E. Miller, Parallel program schemata, J. Comput. Syst. Sci. 3 (1969) 147–195.
[30] D. Kozen, Automata and Computability, Springer, 1997.
[31] S. Kurtz, S. Mahaney, J. Royer, J. Simon, Biological computing, in: Complexity Theory Retrospective II, 1997, pp. 179–195.
[32] M. Lachmann, G. Sella, The computationally complete ant colony: global coordination in a system with no hierarchy, in: F. Morán, A. Moreno, J.J. Merelo, P. Chacón (Eds.), Advances in Artificial Life: Third European Conference on Artificial Life, Springer, 1995, pp. 784–800.
[33] M.R. Lakin, A. Phillips, Modelling simulating and verifying Turing-powerful strand displacement systems, in: L. Cardelli, W. Shih (Eds.), DNA Computing and Molecular Programming, in: Lecture Notes in Computer Science, vol. 6937, Springer, 2011, pp. 130–144.
[34] M.R. Lakin, D. Stefanovic, A. Phillips, Modular verification of chemical reaction network encodings via serializability analysis, Theor. Comput. Sci. 632 (2016) 21–42.
[35] M.R. Lakin, S. Youssef, F. Polo, S. Emmott, A. Phillips, Visual DSD: a design and analysis tool for DNA strand displacement systems, Bioinformatics 27 (2011) 3211–3213.
[36] J. Leroux, Vector addition systems reachability problem (a simpler solution), in: EPiC, vol. 10, 2012, pp. 214–228.
[37] S. Lin, T. Rado, Computer studies of Turing machine problems, J. ACM 12 (1965) 196–212.
[38] M.O. Magnasco, Chemical kinetics is Turing universal, Phys. Rev. Lett. 78 (6) (1997) 1190–1193.
[39] A.R. Meyer, L.J. Stockmeyer, The equivalence problem for regular expressions with squaring requires exponential space, in: IEEE Annual Symposium on Switching and Automata Theory, IEEE, 1972, pp. 125–129.
[40] E.L. Post, Recursive unsolvability of a problem of Thue, J. Symb. Log. 12 (1947) 1–11.
[41] P. Prusinkiewicz, A. Lindenmayer, The Algorithmic Beauty of Plants, Springer-Verlag, 1990.
[42] L. Qian, D. Soloveichik, E. Winfree, Efficient Turing-universal computation with DNA polymers, in: Y. Sakakibara, Y. Mi (Eds.), DNA Computing and Molecular Programming, in: Lecture Notes in Computer Science., vol. 6518, Springer, 2011, pp. 123–140.
[43] L. Qian, E. Winfree, Parallel and scalable computation and spatial dynamics with DNA-based chemical reaction networks on a surface, in: S. Murata, S. Kobayashi (Eds.), DNA Computing and Molecular Programming, in: Lecture Notes in Computer Science., vol. 8727, Springer, 2014, pp. 114–131.
[44] G. Rozenberg, Handbook of Graph Grammars and Computing by Graph Transformation, vol. 1, World Scientific, 1997.
[45] D.B. Searls, Linguistic approaches to biological sequences, Bioinformatics 13 (1997) 333–344.
[46] S.W. Shin, C. Thachuk, E. Winfree, Verifying chemical reaction network implementations: a pathway decomposition approach, Theor. Comput. Sci. 765 (2019) 67–96.
[47] D. Soloveichik, M. Cook, E. Winfree, J. Bruck, Computation with finite stochastic chemical reaction networks, Nat. Comput. 7 (2008) 615–633.
[48] D. Soloveichik, G. Seelig, E. Winfree, DNA as a universal substrate for chemical kinetics, Proc. Natl. Acad. Sci. 107 (2010) 5393–5398.
[49] N. Srinivas, J. Parkin, G. Seelig, E. Winfree, D. Soloveichik, Enzyme-free nucleic acid dynamical systems, Science 358 (2017) eaal2052.
[50] L.J. Stockmeyer, A.R. Meyer, Word problems requiring exponential time (preliminary report), in: Proceedings of the Fifth Annual ACM Symposium on Theory of Computing, STOC, ACM, New York, NY, USA, 1973, pp. 1–9.
[51] A. Tai, A. Condon, Error-free stable computation with polymer-supplemented chemical reaction networks, in: C. Thachuk, Y. Liu (Eds.), DNA Computing and Molecular Programming, in: Lecture Notes in Computer Science, vol. 11648, Springer, 2019, pp. 197–218.
[52] A.J. Thubagere, W. Li, R.F. Johnson, Z. Chen, S. Doroudi, Y.L. Lee, G. Izatt, S. Wittman, N. Srinivas, D. Woods, E. Winfree, L. Qian, A cargo-sorting DNA robot, Science 357 (2017) eaan6558.
[53] R.J. Van Glabbeek, The linear time-branching time spectrum I. The semantics of concrete, sequential processes, in: Handbook of Process Algebra, Elsevier, 2001, pp. 3–99.
[54] E. Winfree, http://www.dna.caltech.edu/ReactionSchemaSimulator/.
[55] W. Yahiro, M. Hagiya, Implementation of Turing machine using DNA strand displacement, in: International Conference on Theory and Practice of Natural Computing, Springer, 2016, pp. 161–172.

Please cite this article in press as: R.F. Johnson, E. Winfree, Verifying polymer reaction networks using bisimulation, Theoret. Comput. Sci. (2020), https://doi.org/10.1016/j.tcs.2020.08.007