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

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Verifying chemical reaction network implementations: A bisimulation approach

Robert Johnson [a,*], Qing Dong [b,1], Erik Winfree [a,c]

[a] Bioengineering, California Institute of Technology, United States
[b] Computer Science, SUNY Stony Brook, United States
[c] Computer Science, California Institute of Technology, United States

ARTICLE INFO

Article history:
Received 28 March 2017
Received in revised form 23 December 2017
Accepted 4 January 2018
Available online xxxx

Keywords:
Chemical reaction networks
Formal verification
Bisimulation
Complexity theory

ABSTRACT

Efforts in programming DNA and other biological molecules have recently focused on general schemes to physically implement arbitrary Chemical Reaction Networks. Errors in some of the proposed schemes have driven a desire for formal verification methods. By interpreting each implementation species as a multiset of formal species, the concept of weak bisimulation can be adapted to CRNs in a way that agrees with an intuitive notion of a correct implementation. The theory of CRN bisimulation can be used to prove the correctness of a general implementation scheme or to detect subtle problems. Given a specific formal CRN and a specific implementation CRN, the complexity of finding a valid interpretation between the two CRNs if one exists, and that of checking whether an interpretation is valid are both PSPACE-complete in the general case, but are NP-complete and polynomial-time respectively under an assumption that holds in many cases of interest. We present effective algorithms for both of those problems. We further discuss features of CRN bisimulation including a transitivity property and a modularity condition, the precise connection to the general theory of bisimulation, and an extension that takes into account spurious catalysts.

1. Background

1.1. Introduction

In molecular programming, many real and abstract systems can be expressed in the language of Chemical Reaction Networks (CRNs). A CRN specifies a set of chemical species and the set of reactions those species can do, and the CRN model allows us to deduce the global behavior of the system from that local specification. CRNs are a useful way to separately analyze the computational and the physical aspects of a system. We can use the CRN model to help analyze real systems [1,2] or design engineered systems [3,4].

The CRN model is particularly useful as a programming language for molecular computation. Small CRNs have been designed to exhibit simple behaviors and to compute simple problems, such as the "rock–paper–scissors oscillator" which oscillates between high concentrations of three species in consistent order [5,6], and the approximate majority network

* Corresponding author.
E-mail address: rfjohnso@caltech.edu (R. Johnson).
[1] Current affiliation: Epic Systems, Madison, Wisconsin.

https://doi.org/10.1016/j.tcs.2018.01.002
0304-3975/

which converts all of two initial populations of species to whichever one was initially greater [7]. Other examples of CRNs designed to compute include a CRN that produces an output with count equal to the larger of two input counts [8] and a CRN that simulates a given Turing machine with arbitrarily small probability of error [9]. To show that using CRNs as a programming language can apply to real molecular systems, Chen et al. experimentally demonstrated an implementation of the approximate majority CRN [3], and Srinivas et al. demonstrated an implementation of the rock–paper–scissors oscillator [4], both using DNA strand displacement cascades [10].

In order to discuss how these CRNs compute, we first define a semantics that specifies the behavior of a CRN, then define computation in terms of, for example, from a given input state, how many of or with what probability a given output molecule is produced. The two most common semantics are *deterministic* or *ordinary differential equation (ODE) semantics*, and *stochastic* or *continuous-time Markov chain (CTMC) semantics*. Deterministic semantics assumes real-valued *concentrations* of species which evolve according to a system of ODEs determined by the reactions. Stochastic semantics assumes integer-valued *counts* of species which transition discretely at random times according to the reactions, with rates based on the counts and the reaction rate constants. Any CRN can be interpreted in either semantics, but its behavior may be slightly or entirely different. In deterministic semantics, for example, the rock–paper–scissors oscillator will oscillate correctly indefinitely, and the approximate majority CRN will correctly convert all molecules to whichever species was initially greater quickly, even as the difference between initial counts approaches zero, except for an equilibrium with both species present when the difference is exactly 0. In stochastic semantics, on the other hand, the rock–paper–scissors oscillator will oscillate for a while but eventually undergo an extinction event and stop oscillating [6], while the approximate majority CRN will still compute correctly with high probability, but may convert all molecules to the wrong species with error probability increasing with smaller initial differences [7]. The CRN from [8] that computes the maximum of its input counts functions identically in deterministic and stochastic semantics, which we suspect is related to the rate-independent property discussed below, a connection which has been partially explored by Chen, Doty and Soloveichik [11]. Finally, the CRN to simulate a Turing machine has only been explored in the stochastic model, since it depends on exact integer counts of certain species; its dynamics in deterministic semantics are probably uninteresting.

There is a certain subclass of CRNs that, while they are interpreted in the stochastic model, do not depend on randomness or relative probabilities of certain trajectories to compute an interesting function. This type of computation has been called "deterministic computation" by some [8], but to avoid confusion with deterministic semantics we refer to it in this paper as *rate-independent computation*. Rate-independent computation requires that the correct answer is produced no matter what reactions fire in what order (subject to a fairness condition), a property which implies that for any choice of positive rate constants, the correct answer is produced with probability 1 in stochastic semantics. Rate-independent computation can compute exactly the semilinear functions [12,8,13], an example of which is computing the maximum of two input counts [8]. A concept of rate-independent computation in the deterministic model has been explored by Chen, Doty and Soloveichik [11], where it can compute exactly positive continuous and piecewise rational linear functions, but the exact relation between the two concepts of rate-independent computation is unknown.

Despite the current progress in CRN computation, there remains a gap between abstract and real CRNs. To illustrate this gap, consider the approximate majority CRN [7,3]:

$$X + Y \xrightarrow{k} 2B$$

$$X + B \xrightarrow{k} 2X$$

$$Y + B \xrightarrow{k} 2Y$$

This abstract CRN quickly and with high probability converts all of the initial X and Y molecules into the same amount of whichever one was initially greater [7]. However, no three molecules with exactly this behavior are known to exist. (In a strict sense, no three molecules with *exactly* this behavior can exist, because for all three reactions to be driven forward would require $X + Y$ to be both lower-energy and higher-energy than $2B$.) For contrast, consider the DNA strand displacement system built by Chen et al. [3] meant to implement this abstract CRN. The DNA system uses additional molecules which are consumed as "fuel" to drive these three reactions, ending up with over 25 each of species and reactions. Without knowing that it is meant to be an implementation of the approximate majority CRN, it might be difficult to tell what the DNA system was meant to do. Even knowing the correspondence, it is not obvious that there is no mistake in that complex implementation.

The issue of verifying correctness is exacerbated by the recent profusion of experimental and theoretical implementations in synthetic biology and molecular programming. Of particular interest to us, Soloveichik et al. [14] designed a systematic way to construct a DNA system to simulate an arbitrary CRN. Since then there have been a number of methods to translate an arbitrary CRN into a DNA strand displacement circuit [14–16]. While each one gave arguments for why it was a correct implementation, they did not come with a general theory of what it means to correctly implement a CRN. In some cases this led to subtle problems, of which we will give examples later. To be certain that such implementations are correct, CRN verification methods were invented. Such methods include Shin's pathway decomposition [17], Lakin et al.'s serializability analysis [18], and Cardelli's morphisms between CRNs [1].

The concept of "bisimulation" was developed as a way of comparing the observable behavior of concurrent systems [19]. Roughly speaking, bisimulation involves coming up with a relation on states such that any action in one of a pair of related

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

states can be matched in the other, leading to another pair of related states. Weak bisimulation, in particular, abstracts away from the details of the system by focusing on "observable" actions and allowing matches between sequences with arbitrary numbers of "silent" actions and one observable action. From this local concept of equivalent states, one can prove global properties of equivalence of the behavior of the systems. Bisimulation has been studied in finite-state systems, Petri nets, and hybrid systems, among others [20–22]. The complexity of bisimulation in finite systems and in Petri nets (which are equivalent to CRNs) has also been studied; particularly relevant to us, whether an (arbitrary) bisimulation relation exists that relates the initial states of two Petri nets is undecidable [21]. However, the direct application of bisimulation to Petri nets used in that result ignores the structure of a Petri net (or CRN) in the following sense: where bisimulation allows matching arbitrary states with each other, CRNs (and equivalently, Petri nets) have a structure on the state space that allows addition of states, and we might require that the bisimulation relation preserves that addition. For example, if $A \cong B$ and $C \cong D$, where $\cong$ is the bisimulation relation, we might require that $A + C \cong B + D$. If such constraints better capture the notion of "equivalent CRNs", they could also be exploited to simplify the tasks of finding a satisfactory bisimulation or proving that none exists.

Motivated by the expectation that there is a natural class of restricted bisimulations that respect the structure of CRNs in a way well-suited for molecular implementations—and that makes analysis tractable—we present a method for comparing an implementation CRN with an abstract CRN based on the concept of bisimulation from concurrency theory [19]. Our method associates each implementation species with a multiset of formal species, then asserts correctness if the reactions reachable from any implementation state are the same as those in the corresponding state in the abstract CRN. Like pathway decomposition [17] and serializability [18] but unlike Cardelli's morphisms [1], our bisimulation method works with the stochastic model for low-copy-number CRNs and doesn't take into account rates or kinetics. Therefore, like pathway decomposition and serializability, bisimulation cannot prove that a property true with some probability in the formal CRN will be true with the same or even approximately the same probability in an implementation CRN, but it can guarantee that any rate-independent behaviors or computations of the formal CRN will carry over to the implementation. The use of interpretations instead of pathways means that some implementations considered correct by pathway decomposition are considered incorrect by bisimulation and vice versa. Interpretations also make bisimulation more local than pathway decomposition or serializability, in that many properties can be checked on individual reactions rather than pathways; we hope this makes bisimulation more understandable and tractable. We show how bisimulation can be used to prove a CRN implementation correct or identify subtle problems. We present an algorithm to check whether a particular interpretation between two CRNs is a bisimulation relation, and an algorithm to find such an interpretation if one exists. We analyze the computational complexity of both problems. We prove that both are PSPACE-complete in the general case but become polynomial time and NP-complete, respectively, when formal reactions are limited to a constant number of reactants. We hope this method can be used in both verifying that engineered systems match their specification and in comparing natural systems to a system simple enough to analyze.

1.2. Related works

Our research into verification of CRN implementations is inspired by a number of works on implementing arbitrary CRNs with DNA strand displacement, such as [14–16]. Soloveichik et al. in [14] present a general construction for DSD implementations of arbitrary CRNs, give an argument that the ODE semantics of their construction should approximate the original CRN in a certain limit, and demonstrate similar (but not identical) behavior on some example CRNs. However, this argument does not address the stochastic model, or rate-independent computation within the stochastic model. (We will show in Section 3.3 that the construction in [14] is in fact correct for rate-independent computation according to our definition of bisimulation, but this is not proven in [14].) Qian et al. in [15] present a "history-free" general CRN implementation suited to the stack machines they then describe, and give an argument for its qualitative correctness in the stochastic model; however the argument is non-rigorous. In fact, the argument in [15] misses an error in the construction as published when applied to certain combinations of reactions, allowing some probability of incorrect behavior when run with low counts of species, which we will discuss further in Section 3.3. Cardelli in [16] presents a simplified, history-free general CRN implementation using only nicked double-stranded gates and single-stranded signals, and defines an algebra which can be used to prove statements about the behavior of such systems. This algebra can prove some desirable properties, but Cardelli in [16] acknowledges that properties such as lack of crosstalk require exploring large state spaces and are thus difficult to prove with that technique.

Visual DSD [23] and Peppercorn [24] provide formal semantics for the behavior of a DNA strand displacement such as the ones mentioned in [14–16]. Both [23] and [24] also provide an algorithm to, given a DSD system, produce a CRN which models its behavior. This allows us to reduce the problem of verifying DSD implementations of CRNs to the problem of checking whether one CRN is a correct implementation of another.

In addition to the bisimulation method we propose, other methods have been proposed for verifying CRN equivalence for the use case of DNA strand displacement implementations of CRNs, usually focusing on rate-independent computation in the stochastic model. Lakin et al. define a method of verification of an implementation CRN that is already divided into one module for each formal reaction [18]. This verification method gives properties of each module individually and of non-interaction between modules that, if satisfied, imply that every trajectory of the implementation CRN is equivalent to a serial execution of trajectories corresponding to some sequence of formal reactions, which is at least sufficient for cor-

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

rectness of rate-independent computation. Shin et al. define a method of computing the "formal basis" of a CRN based on pathways in the implementation CRN that begin and end in formal species [17]. If every pathway can be decomposed into the pathways that make up the formal basis, and certain niceness conditions are satisfied, then this "pathway decomposition" method can guarantee that the implementation CRN will be equivalent, at least in rate-independent computation, to its formal basis. Both of these methods depend on a division of the implementation CRN into "formal" species, "fuel" species, "waste" species, and "intermediate" species. This division is a typical feature of engineered DNA strand displacement implementations of CRNs such as the ones mentioned above, but may not generalize to other types of implementations or to comparing natural systems. Lakin et al. [18] give no algorithm for computationally checking the conditions of their serializability method when they are not already known by design; Shin et al. [17] give an algorithm for finding the formal basis of a CRN, but its complexity is not known and it seems to require more than polynomial space in the general case. Both of these methods have been applied to DNA strand displacement implementations such as those in [14–16]. Where Lakin et al. [18] discuss a method of verification that they apply to a specific class of implementations (two-domain strand displacement), and Shin et al. [17] discuss a method with an algorithm that can be applied to any individual implementation CRN, we hope to present a method with an algorithm that can verify any individual implementation CRN, but is also tractable enough to permit proofs that, for example, any implementation created by a given translation scheme will be correct.

Another area of related work has explored more abstract forms of model reduction and CRN equivalence not shaped by DNA strand displacement CRN implementations. Gay et al. discuss a method meant to be useful for systems biology models, where the exact structure of the network has a number of uncertainties and unknowns [25]. This method is based on two operations, merge and delete, which can be applied to graphs of the reactions of a CRN, such that there is an epimorphism from the detailed graph to the more simple graph if and only if the simple graph is equivalent to the result of some sequence of these rules applied to the detailed graph. Some of these concepts correspond loosely to concepts in bisimulation, for example the merging of two species in [25] is intuitively similar to two implementation species being interpreted as the same formal species, but [25] does not have any equivalent of the requirement in bisimulation that any reaction involving one of two equivalent species has an equivalent reaction that involves the other instead. Possibly for this reason, Gay et al. do not prove any properties of the behavior of their networks that a graph epimorphism implies carry over from one network to another.

In a line more directly related, Cardelli, Tribastone, Tschaikowski, Vandin, and Tognazzi have developed multiple concepts of CRN equivalence based on strong bisimulation. *Forward* and *backward bisimulation* respectively imply two different forms of equivalence of the ODE semantics behavior of the equivalent CRNs [26,27], while *syntactic Markovian bisimulation* implies equivalence of the CTMC semantics [28]. Each of these types of bisimulation is an equivalence relation between species of a single CRN, with properties that imply that a simpler CRN, whose species are the equivalence classes of that relation, has dynamics that are well-defined and equivalent in the appropriate sense to the original CRN. In all three cases it is easily checkable whether a given relation is a (forward, backward, or syntactic Markovian) bisimulation, and the same authors give algorithms and software tools to find such relations [29–31,28]. All three of these methods correspond to strong bisimulation in the sense of [19]: one reaction in one system must be matched by exactly one reaction in the other, and vice versa, as opposed to weak bisimulation, where one reaction in one system can be matched by zero or more reactions in the other. This allows these methods to imply equivalence of kinetics in the deterministic and stochastic models, respectively, which would be much more difficult with weak bisimulation. These methods also use a relation on species to induce the relation on states required by bisimulation in the sense of [19], which disallows phenomena such as a single strand of a DSD circuit representing that one copy of the formal species C and two copies of formal D are all present. Since DSD CRN implementations tend to use both one DSD species representing multiple formal species and one formal reaction taking multiple DSD reactions to implement, none of the DSD schemes presented in [14–16] are equivalent to their formal specifications according to forward, backward, or syntactic Markovian bisimulation.

A preliminary version of this work by the same authors was presented at the DNA22 conference in 2016 [32], which in turn incorporated many elements of Qing Dong's 2012 Masters Thesis [33] on this topic. The present paper has been expanded to include additional examples and improved proofs; an algorithm to check the modularity condition and complexity analysis of that algorithm; an expansion of the theory to handle "spurious catalysts", algorithms for the expanded theory, and complexity analysis of those algorithms; additional examples to illustrate various features of CRN bisimulation; and additional discussion of this theory's place in the field of bisimulation and related verification theories.

2. The chemical reaction network model

We work within the *Chemical Reaction Network* (CRN) model. A CRN is a pair $(\mathcal{S}, \mathcal{R})$, where $\mathcal{S}$ is a finite set of species and $\mathcal{R}$ a finite set of reactions. A *reaction* is itself a triple (R, P, k), where the *reactants* R and *products* P are both multisets of species, and $k > 0$ is a real number. We require that in any reaction $R \neq P$, and that no two reactions (R, P, k_1) and (R, P, k_2) with the same reactants and products exist in the same CRN. Once we define semantics, we will see that these requirements can be met without loss of generality; in both deterministic and stochastic semantics, a reaction (R, R, k) has no effect on the CRN's behavior, and a CRN with two reactions (R, P, k_1) and (R, P, k_2) behaves exactly the same as one where those two are replaced by one reaction $(R, P, k_1 + k_2)$.

We use the notation $\{\!|\ldots|\!\}$ for multisets interchangeably with the chemical notation, e.g. $2A + B$, $\{\!|A, A, B|\!\}$, and $\{\!|2A, B|\!\}$ all refer to the same state. Similarly, we sometimes use the chemical notation for reactions, e.g. $A + B \xrightarrow{k} 2C$ is the same as

$(\{\!|A, B|\!\}, \{\!|2C|\!\}, k)$. The "reversible reaction" notation $A + B \underset{k_2}{\overset{k_1}{\rightleftharpoons}} 2C$ is a shorthand for the two reactions $(\{\!|A, B|\!\}, \{\!|2C|\!\}, k_1)$ and $(\{\!|2C|\!\}, \{\!|A, B|\!\}, k_2)$. Where $S \in \mathbb{N}^{\mathcal{S}}$ is a multiset and $X \in \mathcal{S}$, $S(X)$ refers to the count of X in S; this matches the formal definition of $\mathbb{N}^{\mathcal{S}}$ as the set of functions $\mathcal{S} \to \mathbb{N}$. Multisets can be added and multiplied by scalars componentwise, and can be compared componentwise: $S \leq T \iff \forall_X S(X) \leq T(X)$, and $S < T$ if $S \leq T$ and $S \neq T$. If $S \leq T$ then subtraction $T - S$ is defined componentwise. Set operations involving multisets implicitly treat each multiset as the set of all objects which appear at least once; e.g. $\{\!|1, 1, 2|\!\} \subset \{1, 2, 3\}$ but $\{\!|1, 1, 2|\!\} \not\subset \{1\}$. $S \wedge T$ is the componentwise minimum, $(S \wedge T)(X) = \min(S(X), T(X))$. $S \backslash T = S - (S \wedge T)$ is the removal of T from S: $(S \backslash T)(X) = \max(S(X) - T(X), 0)$. The size of a multiset $S \in \mathbb{N}^{\mathcal{S}}$ is the number of objects in it, $|S| = \sum_{X \in \mathcal{S}} S(X)$.

The CRN model comes with two common interpretations or semantics: deterministic semantics and stochastic semantics. In deterministic semantics, the state of the CRN at any given time is a vector $s \in \mathbb{R}^{\mathcal{S}}_{\geq 0}$ of the concentrations of each species, which evolves according to a set of ordinary differential equations. These differential equations come from the reactions, and for each $X \in \mathcal{S}$, where $r_i = (R_i, P_i, k_i)$, $\frac{ds(X)}{dt} = \sum_{r_i \in \mathcal{R}} \rho(r_i, s)(P_i(X) - R_i(X))$. Here $\rho(r_i, s)$ refers to the "rate" of reaction r_i in state s, which according to *mass-action kinetics* is given by $\rho(r_i, s) = k_i \prod_{Y \in \mathcal{S}} s(Y)^{R_i(Y)}$. Since this paper mostly does not deal with deterministic semantics, we only briefly mention it.

In stochastic semantics, the state of the CRN at any given time is a vector $S \in \mathbb{N}^{\mathcal{S}}$ of the counts of each species, which transitions probabilistically to other states, forming a continuous-time Markov chain. In any given state S, each reaction $r_i = (R_i, P_i, k_i)$ has a *propensity* of firing, which in stochastic mass-action kinetics is $\rho(r_i, S) = k_i \prod_{X \in \mathcal{S}} \frac{S(X)!}{(S(X) - R_i(X))!}$, and the reaction takes the CRN from state S to state $S - R_i + P_i$. Note that this expression is defined if and only if $S \geq R_i$; when $S \not\geq R_i$ we set $\rho(r_i, S) = 0$. These propensities play the role of state transition rates in the continuous-time Markov chain.

In the stochastic model, each possible behavior of a CRN is specified by a timed trajectory: an initial state $S_0 \in \mathbb{N}^{\mathcal{S}}$ together with a (finite or infinite) sequence of reactions $r_i = (R_i, P_i, k_i) \in \mathcal{R}$ and times t_i at which they occur, with $t_i > t_{i-1} > 0$. 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; each trajectory can be identified with the set of all timed trajectories with that initial state and sequence of reactions. A trajectory implicitly specifies a sequence of states $S_i = S_0 + \sum_{j \leq i}(P_j - R_j)$, but a sequence of states is not enough to specify a trajectory. For example, if $A \xrightarrow{k_1} B$ and $X + A \xrightarrow{k_2} X + B$ are both reactions, then the sequence of two states $(S_0, S_1) = (\{\!|X, A|\!\}, \{\!|X, B|\!\})$ does not specify which of those two reactions happened, which is sometimes important. The CTMC implicitly specifies a probability distribution over timed trajectories, and since a trajectory interpreted as a set of timed trajectories will be a measurable set in this probability space, we also get a probability distribution over trajectories. For rate-independent computation, we care only about which trajectories are possible, ignoring the times of the reactions and the relative probabilities. We say that a finite trajectory is *valid* if and only if it has nonzero probability, and an infinite trajectory is valid if every finite prefix is valid. Note that a trajectory is valid if and only if every reaction is possible in the state it occurs, i.e. $R_i \leq S_{i-1}$ for all i. Since whether a trajectory is valid in a CRN does not depend on the rate constants of any reaction (as long as they are all positive), and from here on we are generally working with rate-independent computation, we write reactions as a pair (R, P) or $R \to P$ instead of a triple (R, P, k) or $R \xrightarrow{k} P$. In general when we speak of "the trajectories of a CRN" we mean the valid trajectories.

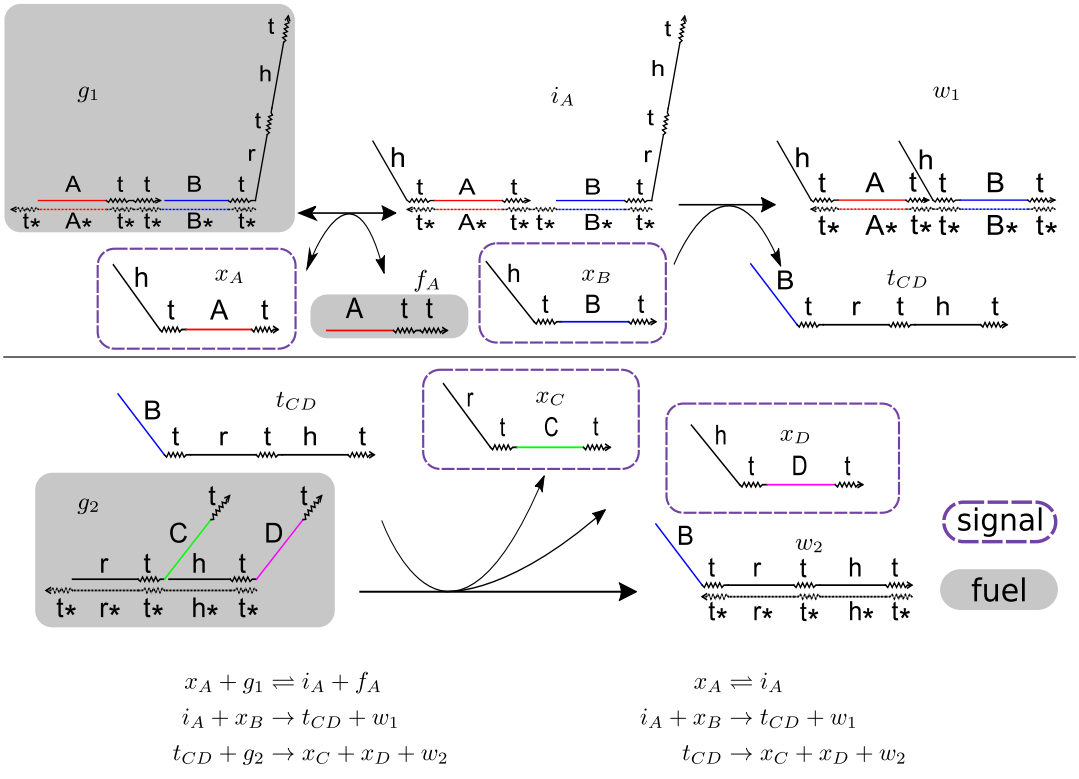
A state T is *reachable* from a state S if T is the result of a valid finite trajectory that starts in S. We say a state T is *coverable* from a state S if there is some $T' \geq T$ such that T' is reachable from S. While the set of reachable states (from any given initial state) is an important aspect of the behavior of a CRN, it does not contain all the information about that CRN. For example, the two CRNs $(\{A, B, C\}, \{A \to B, B \to C, C \to A\})$ and $(\{A, B, C\}, \{A \to C, C \to B, B \to A\})$ have exactly the same set of reachable states T from any given initial state S, but in an external context that distinguishes A, B, and C from each other these two sets of reactions are clearly different in a meaningful way. If however the set of (valid) trajectories of two CRNs are the same, then the two CRNs must be identical: since in particular the length-zero trajectories (i.e. states) are the same, so the sets of species are the same, and the length-one trajectories (single reactions) are the same. We say that two CRNs are *isomorphic* if there is a bijection between the sets of species such that the set of reactions of one, after applying this bijection, equals the set of reactions of the other.

3. The meaning of correctness

3.1. Interpretations

Schemes for translating an arbitrary abstract CRN into a DNA Strand Displacement (DSD) implementation [14–16] provide designs for the necessary DNA molecules, but how these molecules interact is best described by a model of the relevant biophysics. Reaction enumerators such as Visual DSD [23] and Peppercorn [24] produce, given a set of DNA molecules, a description of their predicted interactions as a CRN, allowing us to compare it to the original CRN using the same language. Since most molecular systems can be described as CRNs, defining correctness as a comparison of CRNs will also cover much more general cases, not limited to DNA strand displacement. We refer to the original abstract CRN as the *formal CRN* $(\mathcal{S}, \mathcal{R})$ and the model's enumerated CRN as the *implementation CRN* $(\mathcal{S}', \mathcal{R}')$, which is usually larger than the formal CRN. As a convention, we assume that the formal CRN and the implementation CRN make use of disjoint sets of species. When using

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002



$$x_A + g_1 \rightleftharpoons i_A + f_A \qquad x_A \rightleftharpoons i_A$$
$$i_A + x_B \to t_{CD} + w_1 \qquad i_A + x_B \to t_{CD} + w_1$$
$$t_{CD} + g_2 \to x_C + x_D + w_2 \qquad t_{CD} \to x_C + x_D + w_2$$

Fig. 1. Implementation of $A + B \to C + D$ using the scheme described in [14]. Top: DNA complexes and reactions, given as a diagram of the DNA strand displacement circuit. Each complex shown in the diagram is one species in the enumerated CRN, and arrows are reactions that would be enumerated by a reaction enumerator. Designated "signal" species are enclosed in dashed boxes, and designated "fuel" species in gray boxes. Bottom left: Direct translation of reactions in the implementation CRN. Bottom right: Implementation CRN after removing fuels.

verification to compare a detailed model of a natural system with unknown function to a simpler abstract CRN with known function, the natural system is the implementation and the abstract system is the formal CRN.

Although the definition of correctness we will propose is general, some of its parts are inspired by engineered implementations such as DNA strand displacement. There are three important features typical of engineered implementation CRNs that a concept of correctness must deal with. First, there is typically for each formal species A an implementation species x_A intended to correspond specifically to it, sometimes called a "signal species". Second, certain implementation species must always be present for the system to work, and are designated "fuel species". Fuel species are typically assumed to be held at a constant concentration, for example by setting their initial concentration high enough that it does not vary significantly over the running time of the CRN. In this situation, we can approximate the implementation CRN by a simplified CRN with all fuel species removed; e.g. if g_1 is a fuel, the reaction $x_A + g_1 \to i_A$ can be replaced by $x_A \to i_A$ with no loss of meaning. This approximation holds not just for rate-independent computation, but for both stochastic and deterministic semantics in the following sense: if a species g_1 is (approximately) at a constant concentration c over the time interval considered, the equations introduced in Section 2 for the dynamics of all other species will be (approximately) the same if the reaction $x_A + g_1 \xrightarrow{k} i_A$ is replaced by $x_A \xrightarrow{kc} i_A$ [14]. Third, certain species are inert side products whose further presence or absence has no effect on the correctness of the system behavior, and are thus designated "waste species". Such species can also be removed with no effect on the system dynamics. However, one advantage of our theory is that it does not need to distinguish signal species or waste species from each other or from other species: while knowing that a given species is a signal or a waste can be a helpful hint for finding an interpretation in our theory, it is not necessary, and there are no special rules for signal or waste species. Our theory still requires that fuel species be removed before applying the theory.

Fig. 1 gives an example of this process for the formal reaction $A + B \to C + D$, yielding an implementation CRN with four reactions. (Names such as x_A and t_{CD} are based on the intent of the designers of the CRN, but the subscripts have no theoretical meaning.) The signal species x_A can freely convert to and from i_A, and the strand t_{CD} can produce the signals x_C and x_D (and waste w_2). Intuitively, i_A *is* an A and t_{CD} *is* a C and a D; in this sense the first and third reactions are silent, and the second *is* $A + B \to C + D$. We use this intuition as a basis for our definition of correctness. Formally, we define an *interpretation* of the implementation species (Fig. 2):

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

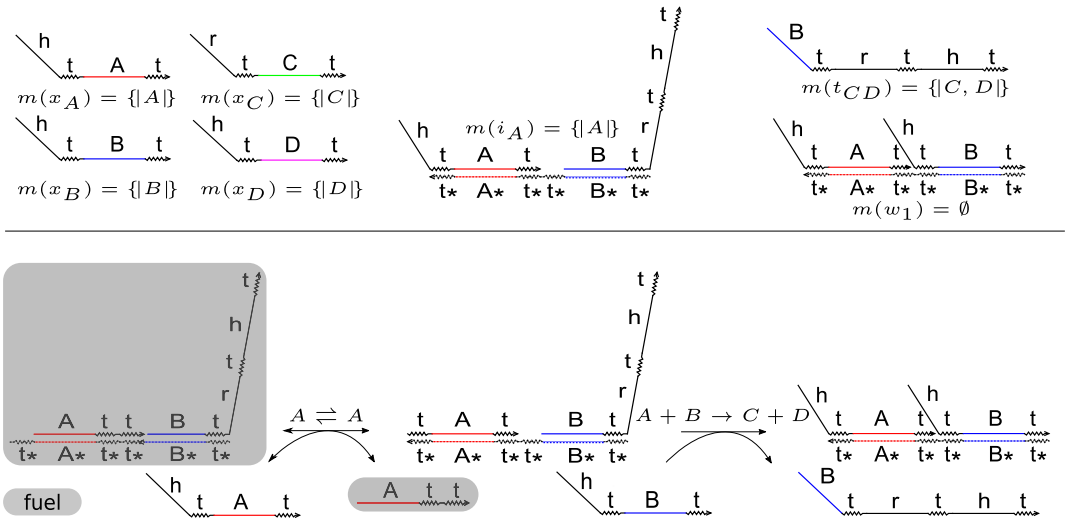


Fig. 2. Interpretation of the implementation CRN in Fig. 1. $m(t_{CD}) = A + B$ would also be a valid interpretation for this CRN.

Definition 3.1. 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: $m(\sum_{i=1}^{n} a_i X_i) = \sum_{i=1}^{n} a_i m(X_i)$. We also define an 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 $m(i_{AB} \to x_A + t_{BC}) = A + B \to A + B + C$.

The interpretation of an implementation reaction is always a pair (R, P) of multisets of formal species, or τ, but (R, P) may not be in $\mathcal{R}$. Any such pair is a *reaction in the language of the formal CRN*, but is a *formal reaction* only if $(R, P) \in \mathcal{R}$. Similarly, (R', P') is an *implementation reaction* only if it is in $\mathcal{R}'$.

In the following notation, S', T', S'', and T'' refer to implementation states; S and T to formal states; r' to an implementation reaction; and r to a reaction in the language of the formal CRN or τ. When a formal reaction r takes state S to state T, we write $S \xrightarrow{r} T$; $S' \xrightarrow{r'} T'$ is similar. Note that if $S \xrightarrow{r} T$, then $r = (R, P) \in \mathcal{R}$ as well as $S - R + P = T$, and analogously for the implementation. Further, we write $S' \xrightarrow{r} T'$ when $S' \xrightarrow{r'} T'$ for some r' with $m(r') = r$, which does not require $r \in \mathcal{R}$ (but does require $r' \in \mathcal{R}'$). Note that if $S' \xrightarrow{\tau} T'$ then $m(S') = m(T')$. One of the core concepts of weak bisimulation is the behavior of the system when we abstract away from trivial reactions. To discuss this behavior, we introduce the relation $S' \overset{r}{\Rightarrow} T'$: we say $S' \overset{\tau}{\Rightarrow} T'$ when S' can reach T' via 0 or more trivial reactions, and $S' \overset{r'}{\Rightarrow} T'$ when $S' \overset{\tau}{\Rightarrow} S'' \xrightarrow{r'} T'' \overset{\tau}{\Rightarrow} T'$. Note that $S' \overset{\tau}{\Rightarrow} S'$ and $S \overset{\tau}{\Rightarrow} S$ are always true. $S' \overset{r}{\Rightarrow} T'$ for $r \neq \tau$ is again defined as $S' \overset{r'}{\Rightarrow} T'$ for some r' with $m(r') = r$. $S \overset{r}{\Rightarrow} T$ for $r \neq \tau$ is defined but trivial: $S \overset{r}{\Rightarrow} T \iff S \xrightarrow{r} T$. When the final state is irrelevant, we sometimes write $S' \overset{r'}{\Rightarrow}$, etc., as appropriate. We say an implementation state S' *can reach* an implementation reaction r' when $S' \overset{r'}{\Rightarrow}$, and we say S' *can implement* a formal reaction r when $S' \overset{r}{\Rightarrow}$. (These definitions are based on notation used by Milner in [19], a connection that will be further discussed in Section 5.1.)

3.2. Three notions of correctness

Our notion of correctness is motivated by the earlier observation that the set of valid trajectories defines equivalence between formal CRNs, and allowing renaming of species defines isomorphism. Applying this notion to an implementation CRN with an interpretation introduces two difficulties. First, due to trivial reactions, the implementation trajectory may involve more steps. This is easily solved by defining the interpretation of an implementation trajectory to remove trivial reactions. Second, and more seriously, the full set of interpreted implementation trajectories may cover the formal trajectories, yet particular implementation trajectories may experience restricted options for alternative paths. Two examples of this are shown in Figs. 3 and 4.

In the first example (Fig. 3), there are two implementation species, x_B and y_B, that are both interpreted as B. Because x_B can do anything in the implementation CRN that B can do in the formal CRN, and because x_A can react to become x_B, the implementation CRN can match any trajectory of the formal CRN using x_B and ignoring y_B. However, an implementation trajectory that reaches—or starts from—y_B cannot proceed, whereas, the formal CRN cannot get stuck. To see that the key issue concerns limiting options, of which getting stuck is just a special case, the reader is encouraged to construct an example where the formal CRN is the one in Fig. 3 augmented by the three reverse reactions, but the implementation CRN

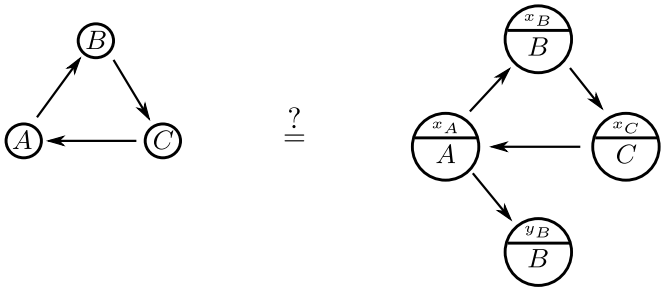


Fig. 3. Example CRNs with the same set of (interpreted) trajectories but different behavior. Circles represent species and arrows represent reactions; implementation species are given with their name above the line and interpretation below the line, so for example x_A is an implementation species with $m(x_A) = \{\!|A|\!\}$. Both CRNs have the same set of trajectories (after interpretation): from any initial count of As, Bs, and Cs, each species can independently change from A to B, B to C, and C to A any finite or infinite number of times. However, the implementation CRN (right) can convert all species to y_B, from which no further reactions are possible, while the corresponding formal state of all Bs can react further.

$$
\begin{array}{lll}
A + B \xrightarrow{1.0} 2B & x_A + x_B \xrightarrow{1.0} 2x_B & y_A + x_B \xrightarrow{1.0} x_B + y_B \\
B + C \xrightarrow{1.0} 2C & x_B + x_C \xrightarrow{1.0} 2x_C & y_B + x_C \xrightarrow{1.0} x_C + y_C \\
C + A \xrightarrow{1.0} 2A & x_C + x_A \xrightarrow{1.0} 2x_A & y_C + x_A \xrightarrow{1.0} x_A + y_A \\
2A \xrightarrow{0.01} 2B & 2x_A \xrightarrow{0.01} 2x_B & x_A + y_A \xrightarrow{0.01} 2y_B \\
2B \xrightarrow{0.01} 2C & 2x_B \xrightarrow{0.01} 2x_C & x_B + y_B \xrightarrow{0.01} 2y_C \\
2C \xrightarrow{0.01} 2A & 2x_C \xrightarrow{0.01} 2x_A & x_C + y_C \xrightarrow{0.01} 2y_A
\end{array}
$$

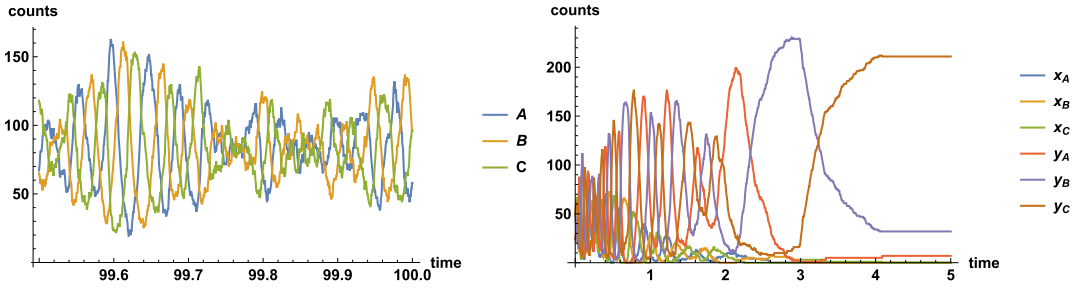


Fig. 4. Modified version of the rock–paper–scissors oscillator [5,6] and an incorrect implementation. Adding the reactions $2A \xrightarrow{0.01} 2B$ etc. ensures that the formal CRN oscillates forever under stochastic semantics (left CRN, left image); without these reactions, eventually the count of one will hit zero and can never be recovered [6]. An implementation CRN with two variants (x_A, y_A, etc.) of each formal species oscillates correctly over short periods of time, and the sets of trajectories of the two CRNs are the same; however, the implementation CRN can and eventually will reach a state where all species are in y form, in which case no further reactions can happen (right CRN, right image).

can become trapped in a subspace where only the "clockwise" trajectories are possible, although it can never get stuck. To appreciate the subtlety of the problem, in our second example (Fig. 4), there are two forms of each formal species, and while the x forms can copy the formal CRN exactly, if all species are converted to y forms in the implementation CRN then no further reaction can happen. As mentioned earlier, this paper is not concerned about differences in kinetics or the probability distribution over trajectories; however, we would like to be able to ensure that properties about what states can be visited in the future, and in what order, are preserved in the implementation. Effectively, the naive definition of trajectory equivalence requires that for every formal state there *exists* an implementation state with the same interpretation and behavior, while we need a finer-grained notion of trajectory equivalence that requires for every formal state, *all* implementation states with the same interpretation have the same behavior. As defined formally below, the finer-grained notion becomes a satisfactory definition of correctness.

Although trajectory equivalence as defined below has the desired meaning, since the sets of trajectories are generally infinite, we would like a more local definition that facilitates efficient computational analysis. We define three local conditions on the interpretation which we show are equivalent to trajectory equivalence. As further evidence that our notion of correctness is sound, we show that these three conditions are equivalent to a special case of weak bisimulation from con-

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

currency theory [19]. (We discuss this connection further in Section 5.1.) This gives us three notions of correctness, given a formal CRN, an implementation CRN, and an interpretation:

Definition 3.2 *(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 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 $S' \overset{r'}{\Rightarrow}$.

III **Weak bisimulation**
- (i) For all implementation states S',
  if $S' \xrightarrow{r} T'$, then $S \overset{r}{\Rightarrow} T$ 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', $S' \overset{r}{\Rightarrow} T'$ and $m(T') = T$.

A few comments may help explain these definitions. It may seem that the second condition for trajectory equivalence supersedes the first, but it does not: for example, the second condition may be satisfied even if there is no implementation state S' that is interpreted as formal state S, whereas the first condition will not be satisfied in that case. In our definition of weak bisimulation, the use of T and T' is in some sense redundant due to the structure of CRNs: the resulting state of a reaction is determined by the initial state and the reaction, so for example if $S \xrightarrow{r} T$ and $S' \overset{r}{\Rightarrow} T'$ then it already must be the case that $m(T') = T$. The definitions of the delimiting and permissive conditions are thus more suited to the structure of CRNs; we gave the definition of weak bisimulation as we did to match the definitions used in concurrency theory [19], and Theorem 3.1 proves that this definition is still equivalent. Because of the connection to bisimulation theory, when $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ is a correct interpretation we often say that m is a bisimulation from $(\mathcal{S}', \mathcal{R}')$ to $(\mathcal{S}, \mathcal{R})$. When the distinction is important, we refer to a bisimulation that satisfies our additional restrictions (i.e., is an interpretation that satisfies the atomic condition) as a "CRN bisimulation", but for most of this paper when not explicitly comparing the different theories of bisimulation we use "bisimulation" to mean "CRN bisimulation".

Theorem 3.1. *The three definitions of correctness, namely trajectory equivalence, the three conditions on the interpretation, and weak bisimulation, are equivalent.*

Proof. We show that trajectory equivalence implies the three conditions formulation; the three conditions imply weak bisimulation; and weak bisimulation implies trajectory equivalence.

Given trajectory equivalence, we prove the three conditions on m. First, for the atomic condition, consider applying condition I.(i) of trajectory equivalence to formal trajectories of length 0, which are just formal states, and in particular formal states $S_A = \{\!|A|\!\}$ for each formal species A. That the set of trajectories are equal implies that there is an implementation trajectory whose interpretation is the (zero-length trajectory) state S_A, i.e. an implementation state S'_A with $m(S'_A) = \{\!|A|\!\}$. Since implementation species cannot be interpreted as fractional or negative formal species, there is some species $x_A \in S'_A$ with $m(x_A) = \{\!|A|\!\}$, satisfying the atomic condition. Second, for the delimiting condition, consider implementation trajectories of length 1, specifically for each implementation reaction $r' = (R', P')$ the trajectory $R' \xrightarrow{r'} P'$. If r' is trivial, that is $m(r') = \tau$, its interpreted trajectory is a zero-length trajectory; if not, its interpreted trajectory is $m(R') \xrightarrow{m(r')} m(P')$, which by trajectory equivalence must be a formal trajectory. For that to be so, $m(r')$ must be a reaction in $\mathcal{R}$, thus satisfying the delimiting condition. And third, for the permissive condition, for every formal reaction $r = (R, P)$ and implementation state S' with $m(S') \geq R$, the trajectory $m(S') \xrightarrow{r} T$, where $T = m(S') - R + P$, is a formal trajectory. By condition I.(ii) of trajectory equivalence, there is an implementation trajectory starting in S' whose interpreted trajectory is $m(S') \xrightarrow{r} T$. (Note that condition I.(i) implies this for *some* S' with $m(S') = S$, but not necessarily for every S'.) To have that interpretation, that implementation trajectory must have some reaction r' with $m(r') = r$ and all other reactions trivial; this is the definition of $S' \overset{r}{\Rightarrow}$, satisfying the permissive condition.

Given the three conditions, we prove weak bisimulation. Given any S' with $m(S') = S$ and $S' \xrightarrow{r'} T'$ where $r' = (R', P')$, by the delimiting condition either $m(r') = \tau$ is trivial or $m(r') = r = (R, P) \in \mathcal{R}$. If trivial, then $m(T') = m(S') = S$ and $S \overset{\tau}{\Rightarrow} S$ is true by convention. If nontrivial, then $r \in \mathcal{R}$; since $S' \xrightarrow{r'}$ we must have $S' \geq R'$, thus $m(S') \geq m(R') = R$, and

$S \xrightarrow{r} T$ (therefore $S \overset{r}{\Rightarrow} T$) where $T = S - R + P$. Since $T' = S' - R' + P'$, $m(T') = m(S') - m(R') + m(P') = T$, satisfying condition III.(i) of weak bisimulation. Given any S, we find an S' with $m(S') = S$ by taking $S' = \sum_A \alpha_A x_A$, where $S = \sum_A \alpha_A A$ and $m(x_A) = \{\!|A|\!\}$ must exist by the atomic condition. Given any such S' with $S \xrightarrow{r} T$ where $r = (R, P)$, by the permissive condition there is some r' with $m(r') = r$ and $S' \overset{r'}{\Rightarrow}$, which is an abbreviation for $\exists_{T'} S' \overset{r'}{\Rightarrow} T'$, which is further an abbreviation for $\exists_{S''} S' \overset{\tau}{\Rightarrow} S'' \xrightarrow{r'} T'$. (Strictly speaking $S' \overset{r'}{\Rightarrow} T'$ means there is some $S' \overset{\tau}{\Rightarrow} S'' \xrightarrow{r'} T'' \overset{\tau}{\Rightarrow} T'$, but since we are choosing an arbitrary T' we can take $T' = T''$.) Then $m(S') = m(S'') = S$ since they are connected by trivial reactions, and where $r' = (R', P')$ with $m(R') = R$ and $m(P') = P$ we have $T' = S'' - R' + P'$ so $m(T') = S - R + P = T$, satisfying condition III.(ii) of weak bisimulation.

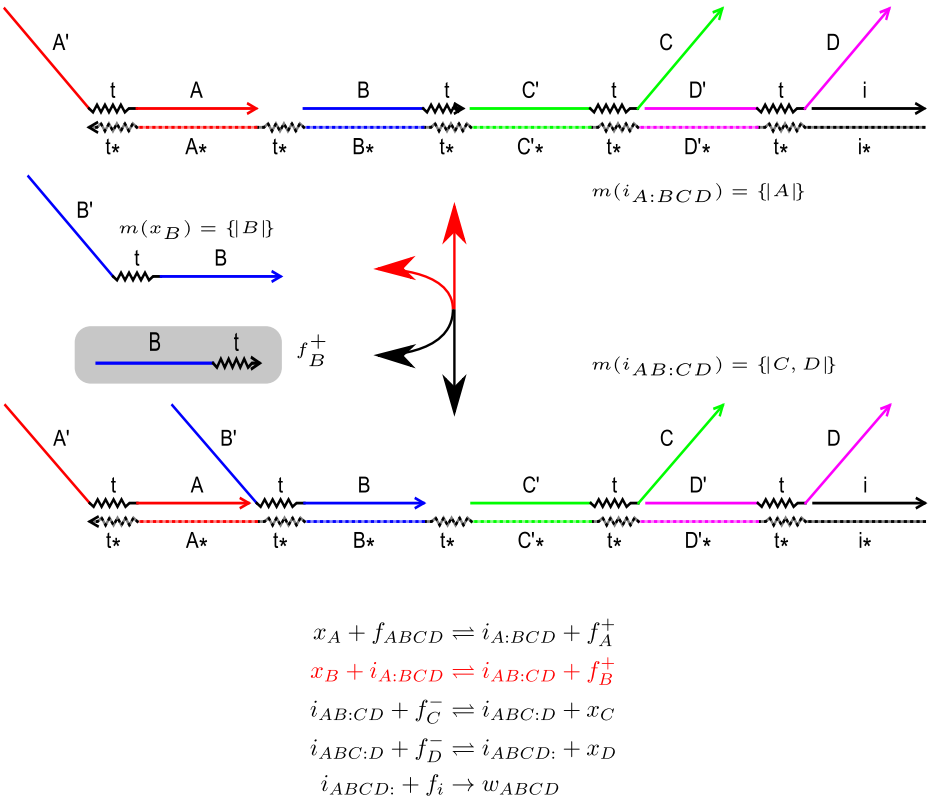
Given weak bisimulation, we prove trajectory equivalence. We first prove condition I.(ii). Given any S'_0 with $S_0 = m(S'_0)$ and any implementation trajectory $(S'_0, r'_1, \ldots, r'_k, \ldots)$ with $r'_k = (R'_k, P'_k)$, let $S'_k = S'_{k-1} - R'_k + P'_k = S'_0 - \sum_{i \le k} R'_i + \sum_{i \le k} P'_k$. Letting $S_k = m(S'_k)$ and $r_k = m(r'_k)$, it follows that either $r_k = \tau$ or else $r_k = (R_k, P_k)$ and $S_k = S_{k-1} - R_k + P_k = S_0 - \sum_{i \le k} R_k + \sum_{i \le k} P_k$ by linearity of m. From bisimulation, since each $S'_{k-1} \xrightarrow{r'_k} S'_k$ we have either $r_k = \tau$ and $S_{k-1} = S_k$, or $r \neq \tau$ and $S_{k-1} \xrightarrow{r_k} S_k$, since for $r \neq \tau$ in the formal CRN $S \overset{r}{\Rightarrow} T \iff S \xrightarrow{r} T$. The interpretation of that implementation trajectory is exactly S_0 followed by those reactions $S_{k-1} \xrightarrow{r_k} S_k$ for which $r_k \neq \tau$, and thus the interpretation is a formal trajectory. Conversely, given S'_0 with $S_0 = m(S'_0)$ and any formal trajectory $(S_0, r_1, \ldots, r_k, \ldots)$ with $r_k = (R_k, P_k)$, letting $S_k = S_{k-1} - R_k + P_k = S_0 - \sum_{i \le k} R_k + \sum_{i \le k} P_k$, we construct an implementation trajectory whose interpretation is that formal trajectory. Given S'_0, define inductively S'_k and r'_k to be an implementation state and reaction such that $S'_{k-1} \overset{r'_k}{\Rightarrow} S'_k$ with $m(r'_k) = r_k$ and $m(S'_k) = S_k$, which exists by condition III.(ii) of weak bisimulation. Expanding each $\overset{r'_k}{\Rightarrow}$ implicitly defines an implementation trajectory $(S'_0, r''_{1,1}, \ldots, r''_{1,l_1}, r'_1, r''_{2,1}, \ldots)$ where each $m(r''_{k,j}) = \tau$ and each $m(r'_k) = r_k$; the interpretation of this trajectory is the formal trajectory $(S_0, r_1, \ldots, r_k, \ldots)$ as desired, proving condition I.(ii). Condition I.(i) follows from condition I.(ii) of trajectory equivalence and condition III.(ii) of weak bisimulation: every implementation trajectory starts from some S' and by condition I.(ii) its interpretation must be a formal trajectory starting from $m(S')$. Conversely, every formal trajectory starts from some S, by condition III.(ii) of weak bisimulation there is some S' with $m(S') = S$, and by condition I.(ii) of trajectory equivalence there is an implementation trajectory starting from S' whose interpretation is that formal trajectory. □

3.3. Applying bisimulation

We now consider how to use bisimulation to analyze our example implementation of $A + B \to C + D$, as shown in Fig. 2. We use the three conditions formulation. The atomic condition is satisfied by the "signal species" x_A, x_B, x_C, and x_D. For the delimiting condition, we check each implementation reaction individually: $i_A + x_B \to t_{CD} + w_1$ is interpreted as $A + B \to C + D$, which is formal, while $x_A \rightleftharpoons i_A$ and $t_{CD} \to x_C + x_D + w_2$ are trivial. The permissive condition says that for every formal reaction and for every implementation state in which that reaction should be able to happen, it can. There is one formal reaction, $A + B \to C + D$, and any state in which it should be able to happen must contain an x_B and either an x_A or i_A, since those are the only species whose interpretations contain either B and/or A. If the state contains x_B and i_A, then the reaction $i_A + x_B \to t_{CD} + w_1$ can happen and satisfies the permissive condition. If the state contains x_B and x_A, then the trivial reaction $x_A \to i_A$ followed by $i_A + x_B \to t_{CD} + w_1$ satisfies the permissive condition.

Now consider a different case. Fig. 5 shows an implementation of $A + B \to C + D$ as described by Qian et al. [15] as a means to implement stack machines, along with a natural interpretation. The species $i_{AB:CD}$ is interpreted as $C + D$, while $i_{A:BCD}$ is interpreted as A and x_B as B. This makes the reaction $i_{AB:CD} \to i_{A:BCD} + x_B$ interpreted as $C + D \to A + B$, which is not a valid formal reaction. Thus the delimiting condition is unsatisfied, and the implementation is not correct according to bisimulation. Although we only show one candidate interpretation, any interpretation will have the same problem provided $m(x_A) = \{\!|A|\!\}$, $m(x_B) = \{\!|B|\!\}$, $m(x_C) = \{\!|C|\!\}$, and $m(x_D) = \{\!|D|\!\}$. The core of the problem in this implementation is that the irreversible step in the pathway happens only after x_C and x_D are released, so there exist trajectories in which one or both is released and then the pathway reverses, producing x_A and x_B again. When analyzed with bisimulation, such a system will (in the closest possible interpretation) have some step that is interpreted as $A + B \to C + D$ but is also reversible, in which case the reverse reaction will be interpreted as $C + D \to A + B$, which is the problem described above. While this does not lead to a problem for the specific construction of deterministic stack machines used in Qian et al. [15], it does identify an error with this as a translation scheme for arbitrary CRNs: if the reaction $A + B \to C + D$ were put together with a reaction $C \to C + E$, then it would be possible to go from $\{\!|A, B|\!\}$ to $\{\!|A, B, E|\!\}$ in the implementation CRN when it is not possible to do so in the formal CRN. As an aside, the inevitability of a reverse reaction interpreted as $C + D \to A + B$ brings up the question of whether this construction would be a correct implementation of a formal CRN with the *reversible* reaction $A + B \rightleftharpoons C + D$. In fact the construction in Fig. 5 would not, since although the delimiting condition would be satisfied, the reaction $C + D \to A + B$ cannot take place starting from $x_C + x_D$, failing the permissive condition: the reaction needs $x_C + x_D + i_{ABCD:}$ to reverse itself, but $i_{ABCD:}$ is not a fuel. However, Qian et al. designed a construction intended to implement reversible reactions, also presented in [15], which when enumerated produces a similar CRN with $i_{ABCD:}$ replaced by a fuel $f_{ABCD:}$ and no $i_{ABCD:} + f_i \to w_{ABCD}$ reaction, and this construction is correct according to bisimulation.

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002



$$x_A + f_{ABCD} \rightleftharpoons i_{A:BCD} + f_A^+$$
$$x_B + i_{A:BCD} \rightleftharpoons i_{AB:CD} + f_B^+$$
$$i_{AB:CD} + f_C^- \rightleftharpoons i_{ABC:D} + x_C$$
$$i_{ABC:D} + f_D^- \rightleftharpoons i_{ABCD:} + x_D$$
$$i_{ABCD:} + f_i \rightarrow w_{ABCD}$$

Fig. 5. The translation scheme from [15], when used as a general CRN implementation, violates the delimiting condition. Species named f are fuels. Top: DNA Strand Displacement diagram of the reversible reaction $x_B + i_{A:BCD} \rightleftharpoons i_{AB:CD} + f_B^+$ with interpretation of involved species given. With the closest possible interpretation (shown), the reverse reaction (highlighted) is interpreted as $C + D \rightarrow A + B$, which violates the delimiting condition when the only formal reaction is $A + B \rightarrow C + D$. Bottom: List of enumerated reactions of the full DSD system for $A + B \rightarrow C + D$, before removing fuels, with the violating reaction $x_B + i_{A:BCD} \rightleftharpoons i_{AB:CD} + f_B^+$ highlighted. Although only one candidate interpretation is shown to be invalid in this figure, any other interpretation either has the same problem or is invalid for other reasons.

A given CRN can be a correct implementation of more than one formal CRN. Given an interpretation, there is only one possible formal CRN for which *that interpretation* might be a bisimulation, but a given implementation CRN may have multiple interpretations to multiple formal CRNs, more than one of which may be correct bisimulations with an appropriate formal CRN. At the extremes, every CRN is an implementation of itself where $m(x) = \{|x|\}$ for all species x is a bisimulation, and every CRN is an implementation of the CRN with 0 species and 0 reactions where $m(x) = \emptyset$ for all x is a bisimulation. As a more interesting example, consider the implementation CRN shown in Fig. 6(A). The CRN has 16 species $x_{i,j}$ for $1 \leq i, j \leq 4$, with reactions $x_{i,j} \rightleftharpoons x_{i+1,j}$ and $x_{i,j} \rightleftharpoons x_{i,j+1}$ for all appropriate i, j. For now, consider a formal CRN with 4 species $\{N, S, E, W\}$ (with unspecified reactions), with the constraint $m(x_{1,1}) = \{|W|\}$, $m(x_{4,1}) = \{|S|\}$, $m(x_{1,4}) = \{|N|\}$, and $m(x_{4,4}) = \{|E|\}$. Fig. 6(B, C, D) shows three possible interpretations for this implementation CRN. The first one is a valid bisimulation for the formal CRN $\{W \rightleftharpoons N, W \rightleftharpoons S, N \rightleftharpoons E, S \rightleftharpoons E\}$; the second is a valid bisimulation for $\{W \rightleftharpoons S, W \rightleftharpoons E, W \rightleftharpoons N\}$. The third interpretation is not a valid bisimulation for any formal CRN: the only formal CRN for which the atomic and delimiting conditions are satisfied is $\{N \rightleftharpoons W, N \rightleftharpoons E, N \rightleftharpoons S, W \rightleftharpoons E, W \rightleftharpoons S, E \rightleftharpoons S\}$, but then the permissive condition is not satisfied for (e.g.) the reaction $E \rightarrow S$ from initial $x_{1,2}$. $m(x_{1,2}) = E$ but $x_{1,2}$ cannot turn into anything that is interpreted as S without passing through something that is interpreted as either W or N, which would be a nontrivial reaction; and the permissive condition requires that a formal reaction be implemented by a sequence of implementation reactions of which all but the last are trivial. The interested reader may enjoy trying to show that there is no interpretation of this implementation CRN which is a valid bisimulation for the formal CRN $\{N \rightleftharpoons W, N \rightleftharpoons E, N \rightleftharpoons S, W \rightleftharpoons E, W \rightleftharpoons S, E \rightleftharpoons S\}$, unless the constraint on the interpretation of the corner species is relaxed.

In many implementations, species with $m(z) = \emptyset$ play a role. We call those species "null species". One type of null species is what theories such as pathway decomposition [17] would call "waste species": implementation species that never appear as a reactant unless all other species involved in the reaction are also waste species. This allows the waste species to be ignored once produced. The species w_1 and w_2 in the example above from Soloveichik et al. [14] (Fig. 1) are examples of this type of null species, which are usually easy to handle with bisimulation.

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

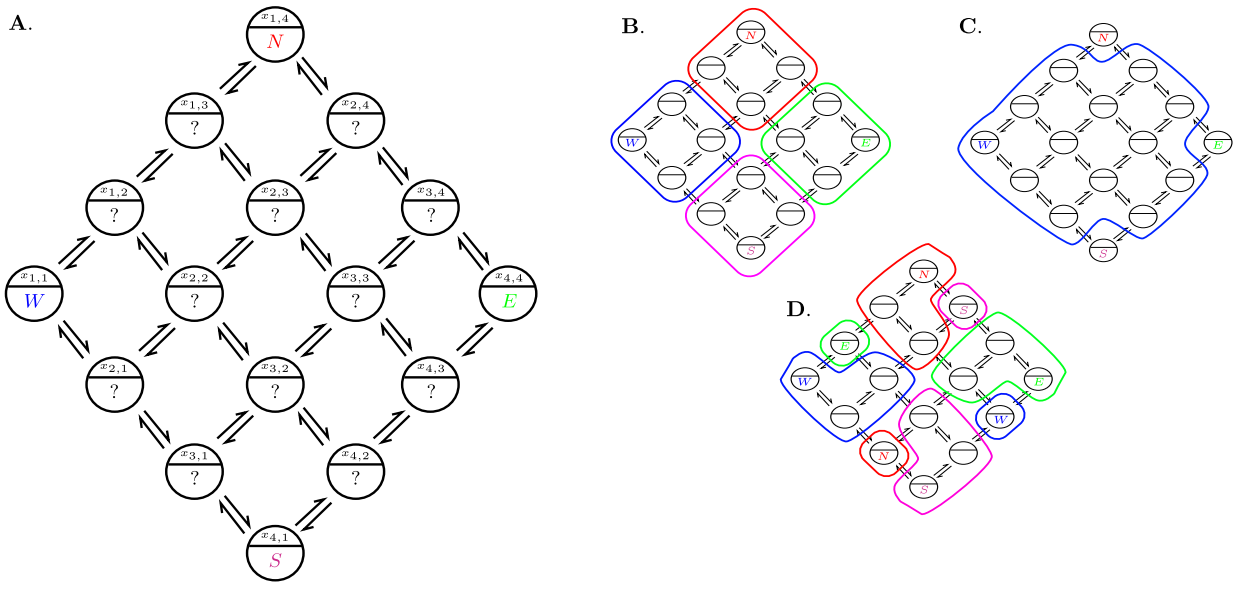


Fig. 6. An example implementation CRN with multiple possible correct interpretations as different formal CRNs. Here each circle represents one implementation species, with the name of the species above the bar and its interpretation below the bar; a "?" means the interpretation is not yet specified. **A.** The implementation CRN with constraints on the interpretation of 4 of its species. **B.**, **C.** Two correct interpretations as two distinct formal CRNs. **D.** An interpretation not correct for any formal CRN. For **B.**, **C.**, and **D.**, species in the same colored circle have the same interpretation.

However, not all null species have to be waste species. Consider the following formal and implementation CRNs:

$$\mathcal{S} = \{A, B\} \qquad \mathcal{S}' = \{x_A, y_A, z, x_B\}$$

$$A \to B \qquad x_A \to y_A$$

$$y_A \to x_A + z$$

$$x_A + 3z \to x_B$$

Let $m(x_A) = m(y_A) = A$, $m(x_B) = B$, $m(z) = \emptyset$. The only reaction which implements $A \to B$ is $x_A + 3z \to x_B$, which requires the null species z. So starting from either x_A or y_A, to implement $A \to B$ the system loops $x_A \to y_A$ and $y_A \to x_A + z$ as many times as needed, then does $x_A + 3z \to x_B$. Since any state whose interpretation has an A must have either x_A or y_A, this proves the permissive condition is satisfied. The atomic condition is satisfied by $m(x_A) = A$ and $m(x_B) = B$, and the delimiting condition is satisfied since the first two implementation reactions are trivial and the third is $A \to B$. Thus this is an example of a correct interpretation with a non-waste null species.

As a final example of the effects of null species, consider a pair of CRNs similar to the previous example:

$$\mathcal{S} = \{A, B\} \qquad \mathcal{S}' = \{x_A, y_A, z, x_B\}$$

$$A \to B \qquad x_A \to y_A + z$$

$$y_A + z \to x_B$$

$$B \to A \qquad x_B \to x_A$$

If we consider the same interpretation as the previous CRN pair, $m(x_A) = m(y_A) = A$, $m(z) = \emptyset$, $m(x_B) = B$, then the permissive condition is not satisfied. In the implementation state $\{\!|y_A|\!\}$, since $m(\{\!|y_A|\!\}) = \{\!|A|\!\}$ the reaction $A \to B$ should be possible, but in the implementation CRN no reactions are possible. (In fact, no correct interpretation exists.) Intuitively, this CRN tried to implement $A \to B$ where the combination $y_A + z$ represents one copy of A; in fact, the theory of pathway decomposition [17] would take this view, with x_A and x_B identified as formal species A and B and with y_A and z considered intermediate species, because pathway decomposition only considers initial states consisting exclusively of formal species when evaluating correct behavior. However, our use of interpretations requires that every implementation species must have a meaning on its own, and counts implementations that rely on combinations having meaning as invalid.

One example of CRN comparison, as discussed in [1], is when a larger CRN contains multiple copies of a smaller CRN, and we want to consider the larger CRN as an implementation of the smaller CRN. As a naive example, consider just two copies of a correct implementation as the implementation CRN:

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

$$\mathcal{S} = \{A, B\} \qquad \mathcal{S}' = \{x_A, y_A, x_B, y_B\}$$

$$A \to B \qquad x_A \to x_B$$

$$y_A \to y_B$$

This example, with $m(x_A) = m(y_A) = A$ and $m(x_B) = m(y_B) = B$, is a correct implementation. However, when we try to do the same thing with bimolecular reactions, problems arise. Consider:

$$\mathcal{S} = \{A, B, C\} \qquad \mathcal{S}' = \{x_A, y_A, x_B, y_B, x_C, y_C\}$$

$$A + B \to C \qquad x_A + x_B \to x_C$$

$$y_A + y_B \to y_C$$

If $m(x_A) = m(y_A) = A$, $m(x_B) = m(y_B) = B$, and $m(x_C) = m(y_C) = C$, then the permissive condition is not satisfied: the state $\{\!|x_A, y_B|\!\}$ is interpreted as $\{\!|A, B|\!\}$, and thus should be able to implement $A + B \to C$, but in fact no reactions are possible and it cannot. This sort of problem is the motivation for the modularity condition which we describe soon, and the reason why checking the permissive condition is difficult, as discussed in Section 4. One solution, in terms of implementation CRN design, is to allow every possible combination of reactants to any reaction to interact:

$$\mathcal{S} = \{A, B, C\} \qquad \mathcal{S}' = \{x_A, y_A, x_B, y_B, x_C, y_C\}$$

$$A + B \to C \qquad x_A + x_B \to x_C$$

$$x_A + y_B \to x_C$$

$$y_A + x_B \to y_C$$

$$y_A + y_B \to y_C$$

This, with the same interpretation as above, is now a correct implementation. However, it can scale poorly: if a formal reaction has k reactants and each one has n possible implementation species, then there are roughly n^k combinations which each need their own reaction. A better-scaling way is to allow different implementation species to interconvert:

$$\mathcal{S} = \{A, B, C\} \qquad \mathcal{S}' = \{x_A, y_A, x_B, y_B, x_C, y_C\}$$

$$A + B \to C \qquad x_A + x_B \to x_C$$

$$y_A + y_B \to y_C$$

$$x_B \rightleftharpoons y_B$$

Again using the same interpretation, this system is a correct implementation. This approach is the one expected by the modularity condition, which helps the scaling behavior.

3.4. Properties of CRN bisimulation

We describe two properties of CRN bisimulation that are likely to be useful when analyzing larger systems. While bisimulation in the classic sense is an equivalence relation between systems [19], our definition of interpretation-dependent CRN bisimulation is a partial order on the set of CRNs. In particular, CRN bisimulation is transitive, which allows us to do complex proofs of correctness in stages. We also show a modularity condition, where the combination of interpretations can be verified using only properties of each individual interpretation. This is particularly useful for general translation schemes where the translation of a whole CRN is the combination of one "module" for each reaction. As an example, we use modularity to prove that the translation scheme in [14] is correct for any CRN.

We first show that CRN bisimulation is transitive. Consider three CRNs: an abstract CRN $(\mathcal{S}, \mathcal{R})$, an implementation CRN $(\mathcal{S}'', \mathcal{R}'')$, and an intermediate CRN $(\mathcal{S}', \mathcal{R}')$. For example, $(\mathcal{S}, \mathcal{R})$ is an abstract CRN, $(\mathcal{S}'', \mathcal{R}'')$ is a low-level reaction enumeration of a prospective DNA implementation of $(\mathcal{S}, \mathcal{R})$, and $(\mathcal{S}', \mathcal{R}')$ is a more high-level reaction enumeration of the same DNA implementation which abstracts away from certain details. Say we have proven that $(\mathcal{S}', \mathcal{R}')$ is a valid implementation of $(\mathcal{S}, \mathcal{R})$ by finding an interpretation $m_1 : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ which is a bisimulation, and similarly have found an interpretation $m_2 : \mathcal{S}'' \to \mathbb{N}^{\mathcal{S}'}$ which is a bisimulation from $(\mathcal{S}'', \mathcal{R}'')$ to $(\mathcal{S}', \mathcal{R}')$. We want to prove that $(\mathcal{S}'', \mathcal{R}'')$, the system we actually have, is a valid implementation of $(\mathcal{S}, \mathcal{R})$, the system we want. The natural interpretation $m : \mathcal{S}'' \to \mathbb{N}^{\mathcal{S}}$ is $m(x) = (m_1 \circ m_2)(x) = m_1(m_2(x))$, treating m_2 as a function of species and m_1 as extended to a function of states. It turns out that this interpretation is in fact a bisimulation.

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

Lemma 3.1 *(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 $(\mathcal{S}, \mathcal{R})$.*

Proof. We use the three conditions formulation of correctness. We refer to $(\mathcal{S}, \mathcal{R})$ as the "formal" CRN, $(\mathcal{S}'', \mathcal{R}'')$ as the "implementation" CRN, and $(\mathcal{S}', \mathcal{R}')$ as the "intermediate" CRN. We show that each condition for m follows from the corresponding conditions for m_1 and m_2.

For any formal species A, by the atomic conditions for m_1 and m_2 there is an intermediate species x_A with $m_1(x_A) = \{\!|A|\!\}$ and implementation species y_A with $m_2(y_A) = x_A$. Then $m(y_A) = m_1(m_2(y_A)) = m_1(\{\!|x_A|\!\}) = \{\!|A|\!\}$, thus m satisfies the atomic condition.

For any implementation reaction $r'' = R'' \to P''$, by the delimiting condition for m_2 its interpretation $m_2(r'')$ is either an intermediate reaction $R' \to P' \in \mathcal{R}'$ or is τ. If $m_2(r'') = \tau$, that means $m_2(R'') = m_2(P'')$ and $m(R'') = m_1(m_2(R'')) = m_1(m_2(P'')) = m(P'')$, so $m(R'' \to P'') = m(r'') = \tau$. If $m_2(r'') = R' \to P'$ is a valid intermediate reaction, then $m(r'') = m_1(R' \to P')$, which by the delimiting condition for m_1 is either a valid formal reaction or trivial.

For any formal state S and reaction r with $S \xrightarrow{r}$ and any implementation state S'' with $m(S'') = S$, that means $S' = m_2(S'')$ is an intermediate state with $m_1(S') = S$. By the permissive condition on m_1, there is some r' with $m_1(r') = r$ and $S' \overset{r'}{\Rightarrow}$. Using the permissive condition on m_2 and the argument used in Theorem 3.1 to show that the permissive condition implies trajectory equivalence, there is a sequence of implementation reactions starting from S'' which implements the intermediate trajectory by which $S' \overset{r'}{\Rightarrow}$. This means that one of those reactions r'' has $m_2(r'') = r'$, some of them interpret via m_2 to various intermediate reactions in that pathway which are trivial under m_1, and the rest of which are trivial under m_2. An implementation reaction trivial under m_2 is trivial under m, as is a reaction which interprets under m_2 to an intermediate reaction trivial under m_1, thus all reactions in this pathway except r'' are trivial under m, so when viewed under m, $S'' \overset{r}{\Rightarrow}$. □

Lemma 3.2 *(Partial order). The following relation is a partial order: $(\mathcal{S}', \mathcal{R}') \gtrsim (\mathcal{S}, \mathcal{R})$ if there exists an $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ which satisfies the atomic, delimiting, and permissive conditions with equality defined as $(\mathcal{S}', \mathcal{R}') \equiv (\mathcal{S}, \mathcal{R})$ if there exists a bijection $n : \mathcal{S}' \to \mathcal{S}$ such that $(n(\mathcal{S}'), n(\mathcal{R}')) = (\mathcal{S}, \mathcal{R})$ where n is extended naturally to sets and reactions.*

Proof. A partial order must be transitive, reflexive, anti-symmetric. Transitivity (if $a \leq b$ and $b \leq c$ then $a \leq c$) follows immediately from Lemma 3.1. Reflexivity ($a \leq a$) is obvious by letting m be the identity function. It remains to show anti-symmetry (if $a \leq b$ and $b \leq a$, then $a = b$), i.e. that given $(\mathcal{S}_1, \mathcal{R}_1)$ and $(\mathcal{S}_2, \mathcal{R}_2)$ with $m_1 : \mathcal{S}_1 \to \mathbb{N}^{\mathcal{S}_2}$ and $m_2 : \mathcal{S}_2 \to \mathbb{N}^{\mathcal{S}_1}$ that each satisfy the atomic, delimiting, and permissive conditions, $(\mathcal{S}_1, \mathcal{R}_1)$ and $(\mathcal{S}_2, \mathcal{R}_2)$ are identical up to a change of species names. The atomic condition implies that $|\mathcal{S}_1| \leq |\mathcal{S}_2|$ and $|\mathcal{S}_2| \leq |\mathcal{S}_1|$, thus the numbers of species are equal and in particular m_1 is a bijection from species in $\mathcal{S}_1$ to sets of exactly one species in $\mathcal{S}_2$ (and the same is true for m_2). To simplify notation, we let $n(x) = y$ if $m_1(x) = \{\!|y|\!\}$; n must be a bijection from $\mathcal{S}_1$ to $\mathcal{S}_2$. (If the CRN has sufficient symmetry, it is not necessarily true that $m_2(n(x)) = \{\!|x|\!\}$, for example if both CRNs are $\{A \to C, B \to C\}$ we could have $m_2(n(A)) = \{\!|B|\!\}$.) Since n is a bijection, any reaction that would be trivial after interpretation (by either m_1 or m_2) must be trivial before interpretation, and thus cannot exist. By the delimiting condition for m_1, every reaction in $\mathcal{R}_1$ must have its image under n in $\mathcal{R}_2$; by the permissive condition for m_1, every reaction in $\mathcal{R}_2$ must have its preimage under n in $\mathcal{R}_1$; thus the two CRNs are equal up to the isomorphism n. □

This result stands in contrast to the definition of bisimulation in transition systems, which is an equivalence relation on states that can be extended to an equivalence relation on systems [19]. We discuss this difference further in Section 5.1.

In Section 3.3 we showed that the translation scheme from [14] is a correct implementation of the single reaction $A + B \to C + D$ according to CRN bisimulation. Intuitively, given a CRN of multiple reactions we should be able to combine the implementations of each such reaction to form a correct implementation of the CRN. In particular, we would like to show that the combined implementation CRN is correct using a condition which we can check on each individual reaction's implementation without having to check any property of the combined CRN. Since, as we will see in Section 4, the time required to check an interpretation scales much worse than linearly in the size of the implementation CRN, such a modularity condition would be a significant saving in the time required. While it is not in general true that combining two correct implementation CRNs gives a correct implementation of the combined formal CRN, there is a modularity condition which guarantees that the combined CRN is correct.

We consider an implementation CRN $(\mathcal{S}_1', \mathcal{R}_1')$ and formal CRN $(\mathcal{S}_1, \mathcal{R}_1)$ with interpretation $m_1 : \mathcal{S}_1' \to \mathbb{N}^{\mathcal{S}_1}$, and another implementation CRN $(\mathcal{S}_2', \mathcal{R}_2')$ and formal CRN $(\mathcal{S}_2, \mathcal{R}_2)$ with interpretation $m_2 : \mathcal{S}_2' \to \mathbb{N}^{\mathcal{S}_2}$, where both m_1 and m_2 are bisimulations. We assume the interpretations are compatible: for each $x \in \mathcal{S}_1' \cap \mathcal{S}_2'$, $m_1(x) = m_2(x)$, which implies $m_1(x) \in \mathbb{N}^{\mathcal{S}_1 \cap \mathcal{S}_2}$. We also assume that the reactions in $\mathcal{R}_1'$ and $\mathcal{R}_2'$ are the only reactions that occur when you combine the implementation species in $\mathcal{S}_1'$ and $\mathcal{S}_2'$; that is, we assume no crosstalk reactions. Whether there is crosstalk can be checked by a reaction enumerator [23,24], but is beyond the scope of this theory. Aside from crosstalk, the main reason for the combined implementation to be incorrect according to bisimulation is a failure of the permissive condition. If some implementation species y in e.g. $\mathcal{S}_1'$ but not in $\mathcal{S}_2'$ has an interpretation that contains a formal species $A \in \mathcal{S}_1 \cap \mathcal{S}_2$, there

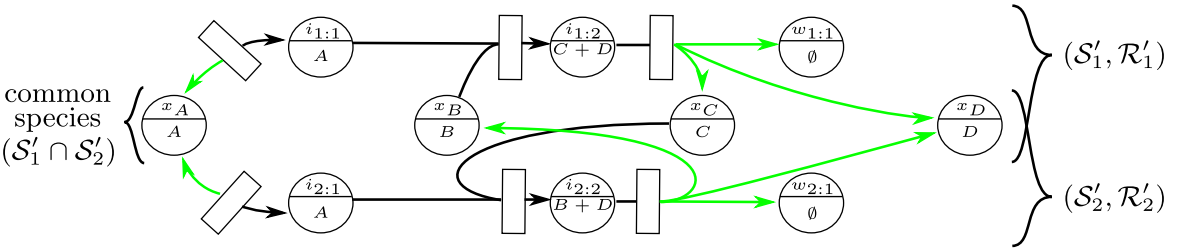


Fig. 7. An implementation CRN that satisfies the modularity condition. Circles represent implementation species with interpretation given below the line, and boxes with arrows represent reactions, with reactants on one side of the box and products on the other; boxes where arrows point both ways are reversible reactions. Here the top CRN $(\mathcal{S}_1', \mathcal{R}_1')$ is a correct implementation of the formal reaction $A + B \to C + D$, and the bottom CRN $(\mathcal{S}_2', \mathcal{R}_2')$ a correct implementation of $C + A \to B + D$. Green arrows (which may appear as light gray in a black-and-white print) indicate reactions used to satisfy the modularity condition in Definition 3.3; for example, $i_{1:2} \overset{\tau}{\Rightarrow} Y + Z$ by reaction $i_{1:2} \to x_C + x_D + w_{1:1}$, where $Y = \{\!|x_C + x_D|\!\} \subset \mathcal{S}_1' \cap \mathcal{S}_2'$ and $Z = \{\!|w_{1:1}|\!\}$ has $m(Z) = \emptyset$. Thus the combined implementation CRN is a correct implementation of the combined formal CRN. If the reverse reaction $i_{1:1} \to x_A$ did not exist, $(\mathcal{S}_1', \mathcal{R}_1')$ would still be a correct implementation of $A + B \to C + D$, but the combined CRN would not satisfy the permissive condition, since state $\{\!|x_C, i_{1:1}|\!\}$ cannot implement $C + A \to B + D$ without that reaction.

may be some formal reaction in $\mathcal{R}_2$ with A as a reactant that cannot be implemented from an implementation state where y is the representation of A, in which case the permissive condition is false. (For example, if in Fig. 7 the reaction $i_{1:1} \to x_A$ was not present, then the interpretation m_1 would still be a correct bisimulation, but the combined interpretation m would fail the permissive condition for where formal reaction $C + A \to B + D$ cannot be implemented from implementation state $\{\!|x_C, i_{1:1}|\!\}$.) If any such species y can, via trivial reactions, "release" any formal species in $\mathcal{S}_1 \cap \mathcal{S}_2$ in its interpretation to implementation species in $\mathcal{S}_1' \cap \mathcal{S}_2'$, then we would think this problem cannot arise. This condition can be checked individually on each module without checking the combined CRN, and we show that this condition guarantees that the combined implementation is correct according to bisimulation. An example of a modular implementation CRN is shown in Fig. 7.

Definition 3.3 *(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 $\{\!|x|\!\} \overset{\tau}{\Rightarrow} Y + Z$ 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 3.2 *(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 $\mathcal{S}_1 \cap \mathcal{S}_2$.*

Proof. We use the three conditions formulation. The atomic condition for m for each formal species A is satisfied by the species x_A that satisfy it for m_1 or m_2, as appropriate, or possibly both; e.g. if $A \in \mathcal{S}_1$ then there is some species $x_A \in \mathcal{S}_1'$ such that $m_1(x_A) = \{\!|A|\!\}$, which implies that $x_A \in \mathcal{S}'$ and $m(x_A) = m_1(x_A) = \{\!|A|\!\}$. Similarly the delimiting condition for m follows from that for m_1 and m_2: for any implementation reaction $R' \to P'$ in $\mathcal{R}'$, that reaction is in either $\mathcal{R}_1'$ or $\mathcal{R}_2'$ (the proof still holds if in both), and its interpretation in m agrees with its interpretation in either m_1 or m_2 as appropriate, which is either a trivial reaction or a formal reaction in $\mathcal{R}_1$ or $\mathcal{R}_2$, which is thus in $\mathcal{R}$.

For the permissive condition, consider a formal reaction $r = R \to P$ and implementation state S' where $R \leq m(S')$. Either $r \in \mathcal{R}_1$ or $r \in \mathcal{R}_2$; without loss of generality say $r \in \mathcal{R}_1$ (where again, the proof still holds if also $r \in \mathcal{R}_2$). Divide S' into species in the first CRN and species not: let $S' = S_1' + S_2'$, where $S_1' \subset \mathcal{S}_1'$ and $S_2' \cap \mathcal{S}_1' = \emptyset$. If $m(S_1') \geq R$, then the permissive condition for m_1 applied to reaction r and state S_1' mean $S_1' \overset{r}{\Rightarrow}$, thus $S' \overset{r}{\Rightarrow}$ by the same sequence of reactions ignoring species in S_2'. In the general case, this means the proof is nontrivial only for formal species in R whose implementations in S' are in S_2', and we need to show that those formal species can be "extracted" into an implementation species in $\mathcal{S}_1'$. This is exactly the modularity condition: for each species $x_i \in S_2'$ there is a sequence of trivial reactions by which $x_i \overset{\tau}{\Rightarrow} Y_i + Z_i$, where $Y_i \subset \mathcal{S}_1'$ and $m(Z_i) \cap \mathcal{S}_1 = \emptyset$. In particular, since $R \to P$ is a reaction in $\mathcal{R}_1$, $R \subset \mathcal{S}_1$ and $R \cap m(Z_i) = \emptyset$. We then have $S' \overset{\tau}{\Rightarrow} S_1' + Y + Z$, where $Y = \sum_i Y_i \subset \mathcal{S}_1'$ and $Z = \sum_i Z_i$. Since $R \cap Z = \emptyset$, $R \leq m(S')$, and $m(S') = m(S_1' + Y) + m(Z)$, we have $R \leq m(S_1' + Y)$. Since $S_1' + Y \subset \mathcal{S}_1'$, the permissive condition for m_1 implies $S_1' + Y \overset{r}{\Rightarrow}$, thus $S' \overset{r}{\Rightarrow}$.

That m is modular with respect to $\mathcal{S}_1' \cap \mathcal{S}_2'$ and $\mathcal{S}_1 \cap \mathcal{S}_2$ also follows simply from the same properties for m_1 and m_2, and the fact that if $m_1(r') = \tau$ or $m_2(r') = \tau$ then $m(r') = \tau$. For any $x \in \mathcal{S}'$, there is a sequence of trivial (under m) reactions by which $x \overset{\tau}{\Rightarrow} Y + Z$, for $Y \subset \mathcal{S}_1' \cap \mathcal{S}_2'$ and $m(Z) \cap \mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$: if $x \in \mathcal{S}_1'$, then such a sequence follows from the modularity of m_1, and if $x \in \mathcal{S}_2'$, from the modularity of m_2. □

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

DNA implementation schemes for arbitrary CRNs such as [14], [15], and [16] typically have a set of common species and for each formal reaction a "module" with additional species and implementation reactions that implement the formal reaction. If the modules have no crosstalk and each one correctly implements its reaction and satisfies the modularity condition, then repeated applications of Theorem 3.2 prove that the entire CRN is a correct implementation.

Corollary 3.3. *Consider a formal CRN $(\mathcal{S}, \mathcal{R})$ with n reactions $\mathcal{R} = \{r_i\}_{i=1}^n$, and n implementation "module" CRNs $(\mathcal{S}'_0 \cup \mathcal{S}'_i, \mathcal{R}'_i)$ with species $\mathcal{S}'_0$ in common, where any $\mathcal{S}'_i$ is disjoint from any $\mathcal{S}'_j$ for $0 \leq i < j \leq n$. If there are interpretations $m_i : \mathcal{S}'_i \to \mathcal{S}$ for $0 \leq i \leq n$ such that the interpretation $(m_0 \cup m_i)$ is a bisimulation from $(\mathcal{S}'_0 \cup \mathcal{S}'_i, \mathcal{R}'_i)$ to $(\mathcal{S}, \{r_i\})$ and is modular with respect to the common implementation species $\mathcal{S}'_0$ and common formal species $\mathcal{S}$, then $m = \bigcup_{i=1}^n m_i$ is a bisimulation from $(\mathcal{S}'_0 \cup \bigcup_{i=1}^n \mathcal{S}'_i, \bigcup_{i=1}^n \mathcal{R}'_i)$ to $(\mathcal{S}, \mathcal{R})$.*

In particular, the translation scheme from [14] discussed earlier satisfies the condition in Corollary 3.3 for $\mathcal{S}'_0 = \{x_A \mid A \in \mathcal{S}\}$, i.e. the signal species. Note that formally, each module contains all signal species—even ones that do not appear in reactions in that module. For example, if the formal CRN has reactions $A + B \to C$ and $A + D \to B$, then the signal species $x_D \in \mathcal{S}'_0$ appears as an implementation species in the module corresponding to $A + B \to C$ but does not appear in any reaction in that module. Although counterintuitive, our theory works fine when some species do not appear in any reactions. Thus Corollary 3.3 proves that for any number of formal reactions, the scheme in [14] produces a correct implementation CRN, as long as the DSD reaction enumerator produces exactly the described reactions and no additional crosstalk reactions.

4. Checking bisimulation

We now have a definition of "correct implementation", and can sometimes prove that a particular implementation is or is not correct. We would like to find a general way to check whether any implementation is correct.

We divide "checking bisimulation" into three questions. First, given a formal and implementation CRN and an interpretation, is the interpretation a bisimulation? Second, if (as in most engineered CRN implementations) we have a formal CRN, implementation CRN, and for each formal species A a designated signal species x_A, is there an interpretation which is a bisimulation and has $m(x_A) = \{\!|A|\!\}$? Finally, given a formal CRN, implementation CRN, and no additional information, is there an interpretation which is a bisimulation?

For complexity purposes, we define the size of a CRN $(\mathcal{S}, \mathcal{R})$ as

$$|(\mathcal{S}, \mathcal{R})| = |\mathcal{S}| + \sum_{X \in \mathcal{S}, R \to P \in \mathcal{R}} (\lceil \log(R(X) + 1) \rceil + \lceil \log(P(X) + 1) \rceil) .$$

This corresponds roughly (up to polynomial factors) to writing the number of species in unary (to cover edge cases involving species not appearing in any reaction), then writing each reaction in the usual chemical notation (e.g. $5X + 3Y \to Z + 2X$). Similarly, for an interpretation $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ we define

$$|m| = \sum_{x \in \mathcal{S}'} \sum_{X \in \mathcal{S}} \lceil \log(m(x)(X) + 1) \rceil ,$$

which corresponds roughly to writing out each interpretation e.g. $m(x_3) = 3A + B$. We find that the complexity of our algorithms is best expressed in terms of three parameters: the *size* $n = |\mathcal{S}| + |\mathcal{R}| + |\mathcal{S}'| + |\mathcal{R}'|$, the total number of species and reactions in the two CRNs; the *arity* $k = \max_{R \to P \in \mathcal{R}} \sum_{X \in \mathcal{S}} R(X)$, the maximum number of reactants in any formal reaction; and the *max stoichiometry* $s = \max_{R \to P \in \mathcal{R} \cup \mathcal{R}', X \in \mathcal{S} \cup \mathcal{S}'} R(X)$. Note that $|(\mathcal{S}, \mathcal{R})| + |(\mathcal{S}', \mathcal{R}')| \leq n^2 \lceil \log(s + 1) \rceil$ and $k \leq s|\mathcal{S}|$, so any algorithm whose time or space complexity is (for example) polynomial in some combination of n, $\log s$, $\log k$, and $|m|$ is (for example) polynomial in $|(\mathcal{S}, \mathcal{R})| + |(\mathcal{S}', \mathcal{R}')| + |m|$. We find that some algorithms are less complex when k is bounded by a constant, such as $k = 2$ limiting the formal CRN to bimolecular reactions, and that the possibility that s is *not* bounded by a constant (in particular, when k is $\omega(n)$) affects the technical details of the proofs but not the result.

4.1. Checking an interpretation

First we consider the problem of, given an interpretation, checking whether it is a bisimulation. We use the three conditions on an interpretation, having proved in Theorem 3.1 that they are equivalent to bisimulation and trajectory equivalence. Given two CRNs and an interpretation between them, the atomic and delimiting conditions are trivial to check. This leaves only the permissive condition.

Checking the permissive condition means, for each formal reaction $r = (R, P)$ and implementation state S' with $m(S') \geq R$, S' can reach via trivial reactions some state from which a reaction that is interpreted as r can happen. Although there are infinitely many such implementation states, we can find a finite set that is sufficient for checking the permissive condition. Fig. 8 shows an example of such a set for the reaction $A + B \to C$ in a hypothetical implementation CRN.

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

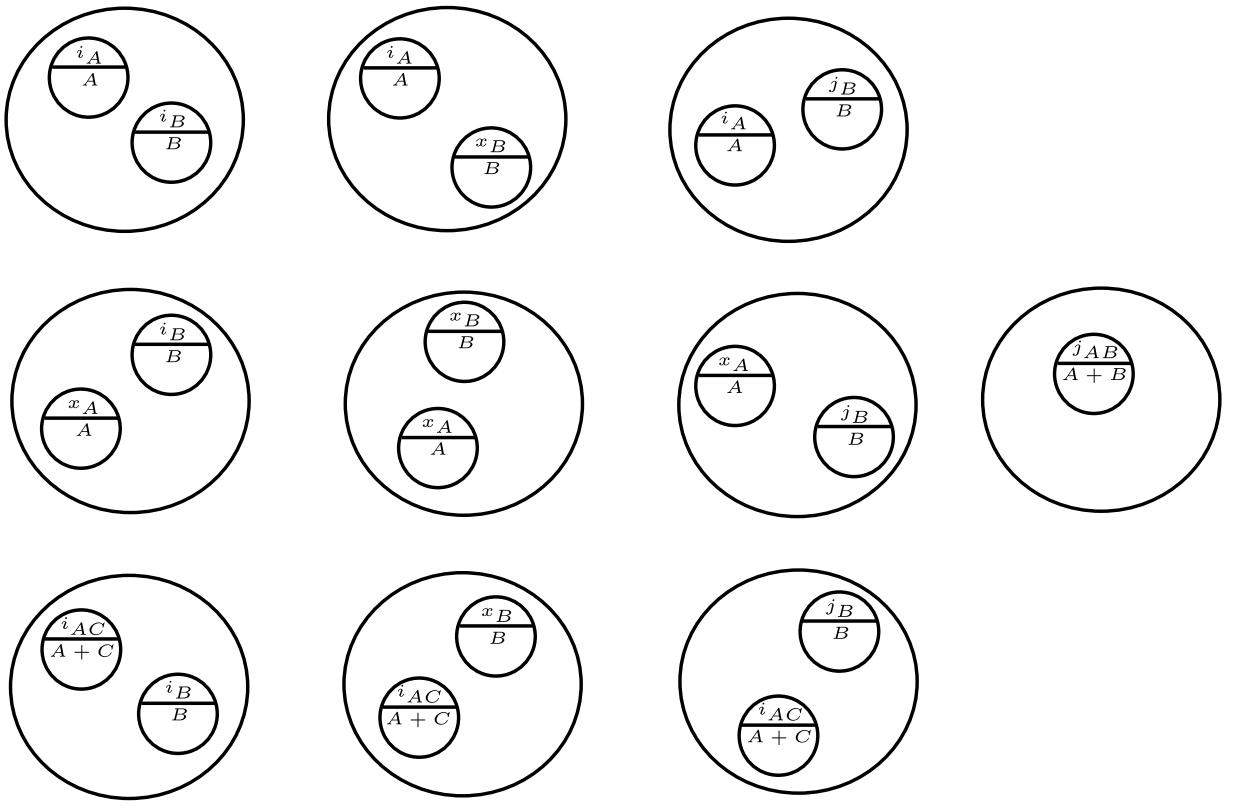


Fig. 8. Example set of minimal implementation states for the reaction $A + B \to C$. Large circles represent minimal states; circles within minimal states represent implementation species, with their interpretation in formal species given below the line. Assuming the 7 implementation species shown are the only species whose interpretation contains either an A or a B, any implementation state whose interpretation contains $A + B$ must contain one of the 10 minimal states shown. The set of minimal states does not depend on the implementation *reactions*, so no reactions are shown.

Definition 4.1. Given a formal reaction $r = (R, P) \in \mathcal{R}$, we say that an implementation state S' is a *minimal implementation state* for r if $m(S') \geq R$ and there is no S'_0 with $S'_0 < S'$ and $m(S'_0) \geq R$. (Equivalently, S' is a minimal element—in the usual sense for partially ordered sets—of the set $\{S' \mid m(S') \geq R\}$, so we often say S' is "minimal for $m(S') \geq R$".) We use the notation $\mathcal{M}(r)$ for the set of minimal implementation states for a formal reaction r.

Lemma 4.1. *Let $(\mathcal{S}, \mathcal{R})$ and $(\mathcal{S}', \mathcal{R}')$ be a formal and implementation CRN with interpretation m. The interpretation m satisfies the permissive condition if and only if, for every formal reaction r and every $S' \in \mathcal{M}(r)$, $S' \overset{r}{\Rightarrow}$.*

Proof. Clearly, if the permissive condition is satisfied, then for any formal reaction r and implementation state $S' \in \mathcal{M}(r)$, $m(S') \xrightarrow{r}$ so by the permissive condition, $S' \overset{r}{\Rightarrow}$. For the converse, assume for every $r \in \mathcal{R}$ and $S' \in \mathcal{M}(r)$ that $S' \overset{r}{\Rightarrow}$, and consider an arbitrary formal reaction $r = (R, P)$ and implementation state S'' with $m(S'') \xrightarrow{r}$, i.e. $m(S'') \geq R$. There is at least one minimal $S' \in \mathcal{M}(r)$ with $S'' \geq S'$, and by assumption $S' \overset{r}{\Rightarrow}$. Since $S'' \geq S'$, the same sequence of reactions can occur in S'', thus $S'' \overset{r}{\Rightarrow}$. Since this is true for every r and S'', the permissive condition is satisfied. □

Lemma 4.2. *Let $(\mathcal{S}, \mathcal{R})$ and $(\mathcal{S}', \mathcal{R}')$ be a formal and implementation CRN with interpretation m, and $r = (R, P) \in \mathcal{R}$ a formal reaction. Let $n = |\mathcal{S}'|$ be the number of implementation species and $k = |R|$ the number of reactants of r. The number of minimal implementation states for r is at most n^k, and all such states can be enumerated in time* $\text{poly}(n^k)$ *and space* $\text{poly}(n, k)$. *When $k \geq n$, in particular the number of minimal implementation states for r is at most $2^{n \log k}$, and they can be enumerated in time $2^{\text{poly}(n, \log k)}$ and space* $\text{poly}(n, \log k)$.

Proof. We describe an algorithm to enumerate all implementation states S' minimal for $m(S') \geq R$ given R, then show that it has the desired complexity and correctly enumerates all minimal states.

If $R = \emptyset$, then the only minimal implementation state for $m(S') \geq R$ is $S' = \emptyset$. If not, choose an arbitrary formal species $A \in R$. For each implementation species x with $A \in m(x)$: Construct a multiset of formal species $Q_x = R \backslash m(x)$ by removing $m(x)$ from R, ignoring any species in $m(x)$ but not in R. Apply this algorithm recursively to enumerate all implementation

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

states S'_x minimal for $m(S'_x) \geq Q_x$. For each such S'_x, the state $S' = S'_x + \{\!|x|\!\}$ has $m(S') \geq R$ and may be minimal. Check if it is minimal by for each $y \in S'$ checking if $m(S' - \{\!|y|\!\}) \geq R$. If none of them are, then S' is minimal; print it and continue enumerating.

We now prove that the algorithm enumerates exactly all S' minimal for $m(S') \geq R$. Since we check whether S' is minimal before printing it, the algorithm clearly does not enumerate any S' with $m(S') \geq R$ that is not minimal. (Without this check, the algorithm could generate a non-minimal implementation state. For example, let $R = \{\!|A, B|\!\}$ where there are implementation species x_1 with $m(x_1) = \{\!|A|\!\}$ and $m(x_2) = \{\!|A, B|\!\}$. The algorithm could first choose A from R and x_1 with $A \in m(x_1)$, generating $Q_{x_1} = \{\!|B|\!\}$, then when called recursively choose B from Q_{x_1} and x_2 with $B \in m(x_2)$, thus generating the state $S' = \{\!|x_1, x_2|\!\}$ which is not minimal, since $S' > \{\!|x_2|\!\}$ and $m(\{\!|x_2|\!\}) = \{\!|A, B|\!\} \geq R$.) That every enumerated S' has $m(S') \geq R$ can be proven by induction on $|R|$. If $|R| = 0$, i.e. $R = \emptyset$, then $m(\emptyset) = \emptyset$ is trivially true. If not, then for each x with $A \in m(x)$ iterated through, $|Q_x| < |R|$ so by assumption each generated S'_x has $m(S'_x) \geq Q_x$. Then the S' generated from that S'_x has $m(S') = m(S'_x) + m(x) \geq Q_x + (m(x) \wedge R) \geq R$. Finally, to prove that every minimal state is enumerated, we again use induction on $|R|$, with the case $|R| = 0$ having only one minimal state, $S' = \emptyset$, which is generated. When $|R| > 0$, consider an arbitrary state S' minimal for $m(S') \geq R$. Where $A \in R$ is the first formal species chosen by the algorithm, there is at least one $x \in S'$ with $A \in m(x)$, and the algorithm at some point iterates through that x. Consider $Q_x = R \backslash m(x)$ as generated in the algorithm, and $S'_x = S' - \{\!|x|\!\}$. If we can show that S'_x is minimal for $m(S'_x) \geq Q_x$, then by assumption the recursive call to the algorithm generates S'_x, thus the algorithm generates $S' = S'_x + \{\!|x|\!\}$. If S'_x is not minimal for Q_x, then there is some $y \in S'_x$ such that $m(S'_x - \{\!|y|\!\}) \geq Q_x$. However, if so, then $y \in S'$ and $m(S' - \{\!|y|\!\}) = m(S'_x - \{\!|y|\!\}) + m(x) \geq Q_x + (R \wedge m(x)) = R$, thus S' was not minimal for R, creating a contradiction. Thus the algorithm generates every minimal S', completing the proof of correctness.

Finally, we prove the algorithm takes time $\text{poly}(n^k)$ and space $\text{poly}(n, k)$. Since the algorithm adds one implementation species at each recursion depth and subtracts at least one species from R at each depth, the depth is at most $|R| = k$. Iterating through at most $|\mathcal{S}'| = n$ implementation species at each depth proves a bound of n^k on the number of minimal implementation states and the $\text{poly}(n^k)$ time bound. At any time the algorithm stores one implementation species plus $\text{poly}(n, k)$ information for the Q_x and S'_x's for each recursion depth, proving the space bound.

If $k \geq n$ then instead of removing one copy of one implementation species x at each recursive step, we choose one implementation species x and a number α, set $Q_{\alpha x} = R \backslash \alpha m(x)$, and mark that x cannot be chosen again in the recursive call. To keep only minimal states, we bound α such that $(\alpha - 1)m(x) \wedge R < \alpha m(x) \wedge R$ (i.e. the αth copy of x is not redundant); since we only choose x with $|m(x)| \geq 1$ and $|R| = k$ this implies $\alpha \leq k$. This algorithm has a depth of at most n, making a choice out of k possibilities at each step and keeping track of at most n numbers each bounded by k, which proves the bounds of time $2^{\text{poly}(n, \log k)}$ and space $\text{poly}(n, \log k)$ given when $k \geq n$. (Note that $k \geq n \geq 2 \Rightarrow n \log k \leq k \log n$, so the bounds given for $k \geq n$ are tighter than the bounds given for the general case.) □

Consider applying the algorithm described in Lemma 4.2 to the example implementation CRN in Fig. 8. Here $R = \{\!|A + B|\!\}$, and assume A is chosen first. The algorithm then splits into different branches based on the choice of implementation species that contains A, where each branch may enumerate one or more minimal states; we discuss the branches separately. One branch of the algorithm will choose j_{AB} to contain A, and stop, since $m(j_{AB}) \geq R$, enumerating one minimal state. Other branches will choose each of i_A, x_A, and i_{AC}. Each of those branches will recursively call the algorithm with $Q_x = \{\!|B|\!\}$; in particular, in the i_{AC} line, $\{\!|A + B|\!\} \setminus \{\!|A + C|\!\} = \{\!|B|\!\}$. Within each of those three branches, the recursive call will again split into branches, with one branch that considers each of i_B, x_B, j_B, and j_{AB} to contain B. The branches that consider j_{AB} will conclude that the resulting minimal states e.g. $\{\!|x_A + j_{AB}|\!\}$ are not minimal, but the other branches will conclude that the resulting states are minimal, enumerating the other 9 minimal states shown in Fig. 8.

Now we have reduced an infinite number of possible initial implementation states to a finite number of minimal implementation states for each formal reaction. We have to check, for each $S'_0 \in \mathcal{M}(r)$, whether $S'_0 \xRightarrow{r}$; equivalently, whether $S'_0 \xRightarrow{\tau} T'$ where $T' \geq R'$ for some implementation reaction with $m(R' \to P') = r$. Checking this for *one* S'_0 is the "superset reachability" or "covering" problem, which was proven by Rackoff [34] and Lipton [35] to be EXPSPACE-complete. Surprisingly, we found that the requirement that *every* $S'_0 \xRightarrow{r}$ makes the permissive condition checkable in PSPACE, by ruling out complex constructions such as Lipton's proof of hardness in [35].

Intuitively, we will show that if some $S'_0 \xRightarrow{r}$ but requires the full complexity of exponential space to determine that, then that complexity will force some other S'_1 with $m(S'_1) \geq R$ to have $S'_1 \not\xRightarrow{r}$. Fig. 9 provides a simplified example of the principle we use. Here we have two minimal states for the formal reaction $r = A + B \to C$. One of them, $S'_0 = \{\!|x_A, x_B|\!\}$ can implement r via the reactions $x_B \to y_B + z$ and $x_A + y_B + z \to x_C$. In doing so, it passes through a non-minimal state $\{\!|x_A, y_B, z|\!\}$, and requires the extra species z to finish implementing r. However, that the extra z is required to implement r means that the other minimal state $S'_1 = \{\!|x_A, y_B|\!\}$ has no way to implement r; so the permissive condition is false, and we don't need to check whether $S'_0 \xRightarrow{r}$. This idea turns out to be generalizable, and allows us to mostly ignore "null species" with $m(x) = \emptyset$, which among other things prevents the complexity necessary for Lipton's proof in [35]. In particular, we will show that when checking for a pathway by which $S' \xRightarrow{r}$ we need only consider the minimal states $\mathcal{M}(r)$ plus a small amount of additional information, and that this can be done in $\text{poly}(n^k)$ time and $\text{poly}(n, k)$ space, where $n = |\mathcal{S}'| + |\mathcal{R}'|$ and $k = |R|$.

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

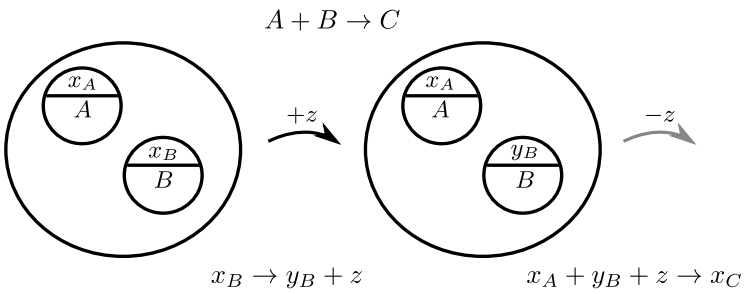


Fig. 9. Minimal state $S'_0 = \{\!|x_A, x_B|\!\}$ can implement $A + B \to C$ by producing and then consuming a null species z. Because z must be produced from x_B, minimal state $S'_1 = \{\!|x_A, y_B|\!\}$ cannot implement $A + B \to C$.

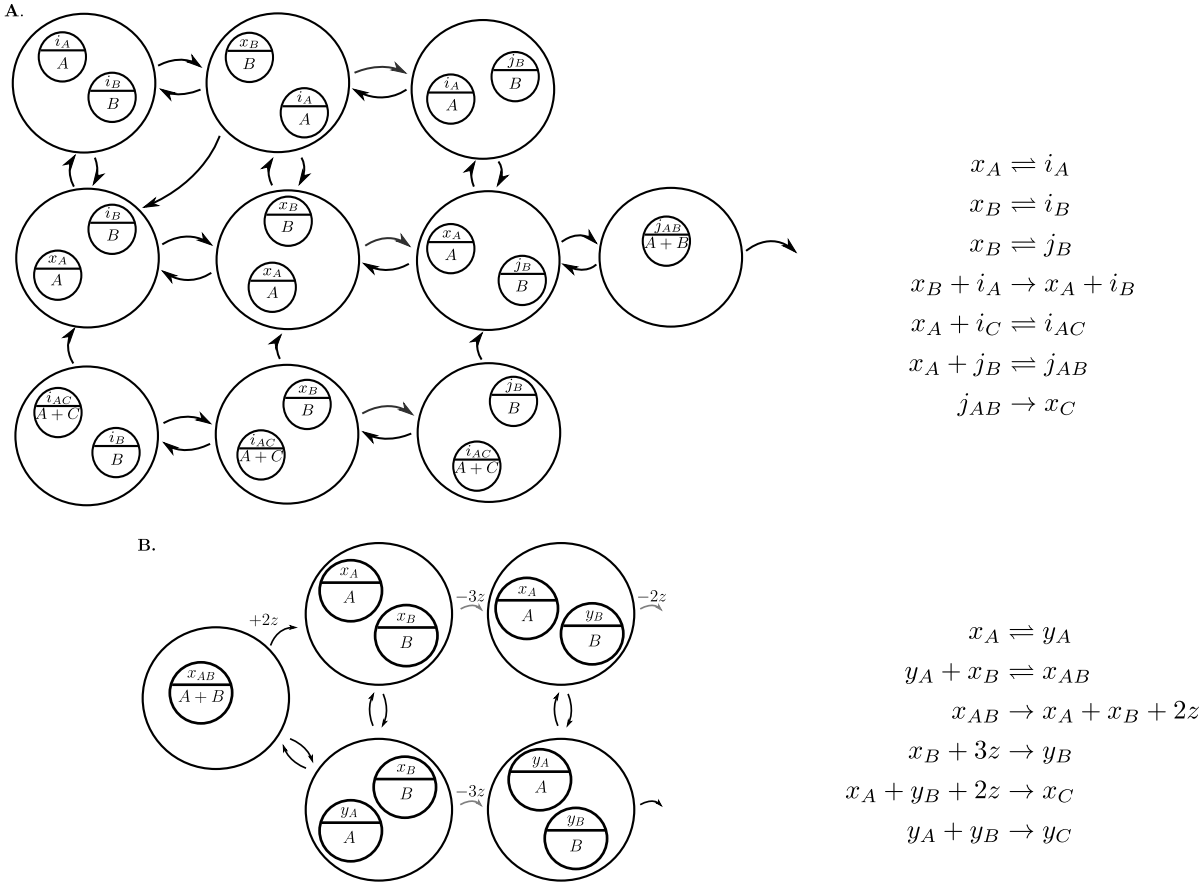


Fig. 10. Example graphs of minimal states. We draw an arrow from a state S'_1 to a state S'_2 if there is a trivial reaction that can occur in S'_1 (plus some null species) and the resulting state is $\geq S'_2$. Arrows "out" (with no target) represent implementation reactions interpreted as the formal reaction in question and that can occur in the minimal state in question plus some null species. **A.** An example graph for the reaction $A + B \to C$ in an implementation CRN without null species, producing the same set of minimal states shown in Fig. 8. Here the permissive condition is true for $A + B \to C$ if and only if every minimal state has a path to some arrow out, which we can see is true for this graph. Note that the reaction $i_{AC} \to x_A + i_C$ in state $\{\!|i_{AC} + x_B|\!\}$ would result in $\{\!|x_A + x_B + i_C|\!\}$, which is not minimal but is $> \{\!|x_A + x_B|\!\}$, so the arrow is from $\{\!|i_{AC} + x_B|\!\}$ to $\{\!|x_A + x_B|\!\}$. Since the reverse reaction $x_A + i_C \to i_{AC}$ is not possible in any minimal state and the feature our algorithms exploit is that we do not need to remember information beyond the minimal state and the null species, as far as we are concerned the reverse reaction is impossible. **B.** An example graph for the reaction $A + B \to C$ in an implementation CRN with one null species z. Here arrows for reactions that consume and/or produce null species are marked with the number of null species they consume and/or the number they produce. Arrows for reactions that consume null species are grayed out, since they cannot occur in the minimal state in question, but may be relevant if the required null species are produced. Checking the permissive condition for $A + B \to C$ in this CRN may require more complex techniques.

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

For a simple case, consider an implementation CRN with no species x where $m(x) = \emptyset$, such as the one shown in Fig. 10A, and consider its graph of minimal states for a formal reaction $r = R \to P$. If, for every minimal state, there is a path through the graph of minimal states to a reaction that implements r, then the permissive condition is true. In fact, the permissive condition is true if and only if such a path exists for every minimal state. If S'_0 is minimal and $S'_0 \overset{\tau}{\Rightarrow} S''$ which is not minimal, then there is some minimal S'_1 with $S'' > S'_1$, which without null species must have $m(S'_1) < m(S'') = m(S'_0)$. Here either $S'_1 \overset{r}{\Rightarrow}$ and therefore $S'_0 \overset{r}{\Rightarrow}$ by reaching S'' and then following the same path by which $S'_1 \overset{r}{\Rightarrow}$, or $S'_1 \overset{r}{\Rightarrow}$ and the permissive condition is false regardless of whether $S'_0 \overset{r}{\Rightarrow}$. Since $m(S'_1) < m(S'_0)$, the path by which $S'_1 \overset{r}{\Rightarrow}$ cannot pass through S'_0, so reducing the question of $S'_0 \overset{r}{\Rightarrow}$ to $S'_1 \overset{r}{\Rightarrow}$ is valid. Effectively, for the purpose of checking the permissive condition, we can pretend S'' is S'_1, thus reducing our search for trajectories to a search for paths through the set of minimal states. Where k is the number of reactants in r and n the number of implementation species, we know from Lemma 4.2 that the number of minimal states is at most n^k when $k \leq n$ and at most $2^{n \log k}$ when $k \geq n$, both of which are exponential in the size of the CRN as defined at the beginning of this section. Because searching for paths through a graph can be done in space logarithmic in the size of the graph [36], we can check the permissive condition in polynomial space when there are no null species. To generalize this, we show that null species and loops do not make this bound worse.

Now consider an implementation CRN with null species, such as the one shown in Fig. 10B, and its graph of minimal states for a formal reaction $r = R \to P$. We can try to apply the same logic as in the case without null species: if a minimal state $S'_0 \overset{\tau}{\Rightarrow} S''$ non-minimal with a minimal state $S'_1 < S''$, either $S'_1 \overset{r}{\Rightarrow}$ and we can pretend S'' is S'_1, or $S'_1 \overset{r}{\not\Rightarrow}$ and the permissive condition is false anyway. Without null species this was valid because $S'' > S'_1$ implies $m(S'') > m(S'_1)$, and thus S'_1 cannot reach S'_0 via trivial reactions. With null species, on the other hand, it may be that $S'' - S'_1$ contains only null species and it may be that $S'_1 \overset{\tau}{\Rightarrow} S'_0$; in particular it may be that both $S'_0 \overset{r}{\Rightarrow}$ and $S'_1 \overset{r}{\Rightarrow}$, but the only path by which $S'_1 \overset{r}{\Rightarrow}$ goes through S'_0 and the only path by which $S'_0 \overset{r}{\Rightarrow}$ goes through S'', creating a loop that will not be found when searching through the graph of minimal states. In fact, this is exactly the case in the CRN shown in Fig. 10B, for example with $S'_0 = \{\!|x_{AB}|\!\}$ and $S'_1 = \{\!|x_A + x_B|\!\}$. All minimal states in that CRN can in fact implement $A + B \to C$, but doing so for e.g. $\{\!|x_A + x_B|\!\}$ requires a "loop" through $\{\!|y_A + x_B|\!\}$ and $\{\!|x_{AB}|\!\}$ to $\{\!|x_A + x_B + 2z|\!\}$, eventually producing enough z for the reactions $x_B + 3z \to y_B$ and $x_A + y_B + 2z \to x_C$, which is interpreted as $A + B \to C$. Since any state with z is a non-minimal state, that path cannot be found by searching through only the graph of minimal states.

In the CRN in Fig. 10B, the path by which $\{\!|x_A + x_B|\!\} \overset{r}{\Rightarrow}$ involves a "loop" by which $\{\!|x_A + x_B|\!\} \overset{\tau}{\Rightarrow} \{\!|x_A + x_B|\!\} + \{\!|2z|\!\}$, i.e. a minimal state can reach a state equal to itself plus some null species. In such a case, that loop can be repeated any number of times to produce any number of z, and when searching for a path, there is no need to keep track of the exact number of z produced: either there are no z, or there are "enough" z produced by a previous loop. Recall the argument we tried to use that failed: if $S'_0 \overset{\tau}{\Rightarrow} S'_1 + Y$, with S'_0 and S'_1 minimal, then either $S'_1 \overset{r}{\Rightarrow}$ and so does S'_0, or $S'_1 \overset{r}{\not\Rightarrow}$ and the permissive condition is false. Recall that this argument only failed because it may be that $S'_1 \overset{r}{\Rightarrow}$ but only by passing through S'_0, in a situation similar to the loop in Fig. 10B. This suggests, which we will show is true, that this example is general: if $S'_0 \xrightarrow{\tau} S'_1 + Y$ for $S'_0, S'_1 \in \mathcal{M}(r)$, then for each $y \in Y$, either y can be completely ignored when checking the permissive condition, or else the following all hold: $m(y) = \emptyset$, there is some $S'_j \in \mathcal{M}(r)$ such that $S'_j \overset{\tau}{\Rightarrow} S'_j + y + \ldots$, and y is only "relevant" after it has been produced in some such loop.

The above discussion allows us to define a graph, which can be both enumerated and searched through in polynomial space, such that paths through the graph correspond to paths by which a given minimal state implements a formal reaction r. The states of this *loopsearch graph* are triples of the form (S', S'_0, ζ) where ζ maps each null species in $\mathcal{Z}$ to 0, 1, or ∞: in each state we are at or covering one minimal implementation state $S' \in \mathcal{M}(r)$, in the middle of a loop beginning at some other state $S'_0 \in \mathcal{M}(r)$, and each null species in $\mathcal{Z}'$ is either absent, produced previously in the current incomplete loop, or present in infinite copies from a previous loop. An example of such a graph is given in Fig. 11.

We use the following notation in defining and discussing the loopsearch graph, in the context of a given formal and implementation CRN with interpretation and a specific formal reaction r. Let $S'_0, S'_1 \in \mathcal{M}(r)$, $\zeta \in 3^{\mathcal{Z}}$ and $Z \in 2^{\mathcal{Z}}$. We write $\zeta^{-1}(x) = \{z \in \mathcal{Z} \mid \zeta(z) = x\}$ for $x \in \{0, 1, \infty\}$; in particular, $\zeta^{-1}(\infty)$ is the set of null species that have been produced in previous loops, and are thus "available" for use later. We write $S'_0 \xrightarrow{\zeta} S'_1 + Z$ if there is a trivial reaction that, for some $n \in \mathbb{N}$, can occur in $S'_0 + n\zeta^{-1}(\infty)$, where the resulting state is some S'' containing S'_1 and at least one copy of each null species in Z. Z may be empty, in which case $S'_0 \xrightarrow{\zeta} S'_1$ is the same as $S'_0 \xrightarrow{\zeta} S'_1 + \emptyset$. Following the terminology of [34], to "cover" a state S in a CRN is to be in a state containing S plus possibly some other species.

Definition 4.2. Given $(\mathcal{S}, \mathcal{R})$ and $(\mathcal{S}', \mathcal{R}')$ a formal and implementation CRN, $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ an interpretation, and r a formal reaction, we define the *loopsearch graph* for r. The loopsearch graph is a directed graph with vertex set $\mathcal{M}(r) \times \mathcal{M}(r) \times 3^{\mathcal{Z}}$, where $\mathcal{Z} = \{z \in \mathcal{S}' \mid m(z) = \emptyset\}$, with some vertices designated as "terminal". Here a vertex (S', S'_0, ζ) is interpreted as, "we are covering state S', in the middle of a loop starting and ending at S'_0, with null species present or absent as determined by ζ", except that $S' = S'_0$ is interpreted as "not in the middle of a loop". $\zeta \in 3^{\mathcal{Z}}$ maps each null species z to $\{0, 1, \infty\}$, a coarse-grained representation of the number of copies of z: we only need to remember whether z is not present ($\zeta(z) = 0$),

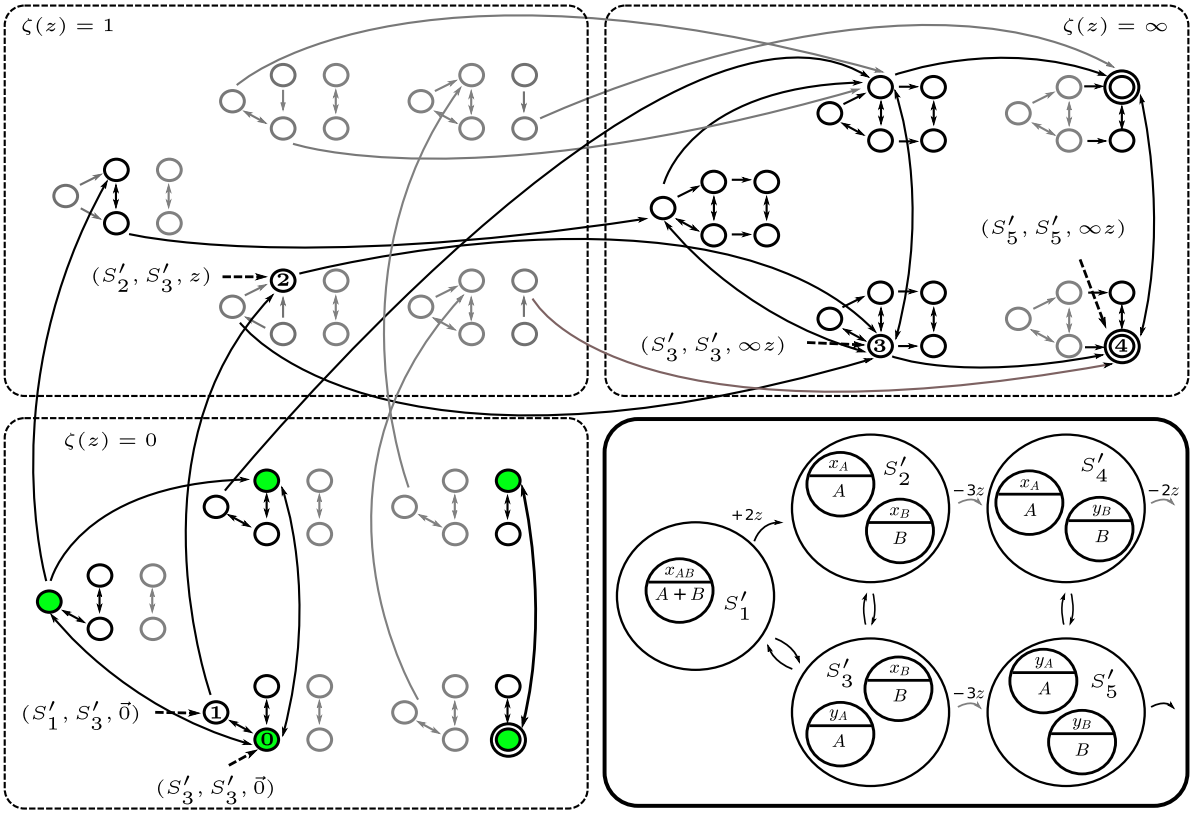


Fig. 11. Example of the loopsearch graph for the formal reaction $A + B \to C$ with implementation CRN from Fig. 10B. The graph of minimal states from Fig. 10B is reproduced at bottom right, with each minimal state given a name S'_i. In the loopsearch graph, initial vertices $(S'_i, S'_i, \vec{0})$ are filled in green (which may appear as light gray in a black-and-white print), and terminal vertices represented by doubled circles. Vertices and edges not reachable from any initial vertex are grayed out, as they are not relevant to the theory or algorithms that follow. The permissive condition is true for $A + B \to C$ if and only if for each initial vertex, there is a path in the loopsearch graph to some terminal vertex. One such path is given by the numbered vertices 0 through 4 from initial vertex $(\{\!|y_A + x_B|\!\}, \{\!|y_A + x_B|\!\}, \vec{0})$ to terminal vertex $(\{\!|y_A + y_B|\!\}, \{\!|y_A + y_B|\!\}, \infty z)$; observe that each of the other four initial vertices can also reach a terminal vertex, so the permissive condition is true for this interpretation.

produced during the current loop (1), or produced as many times as necessary in a previous loop (∞). This interpretation suggests the definition of the edges of the loopsearch graph, which is as follows:

- **Reactions outside a loop:** Whenever $S' \xrightarrow{\zeta} S'_1$ and $\zeta^{-1}(1) = \emptyset$, there is an edge from (S', S', ζ) to (S'_1, S'_1, ζ).
- **Reactions inside a loop:** Whenever $S' \xrightarrow{\zeta} S'_1 + Z$, there is an edge from (S', S'_0, ζ) to (S'_1, S'_0, ζ') for each $S'_0 \neq S'_1$, where ζ' is defined as follows:
  - If $\zeta(z) = 1$ or ∞ then $\zeta'(z) = \zeta(z)$.
  - If $\zeta(z) = 0$ and $z \notin Z$ then $\zeta'(z) = 0$.
  - If $\zeta(z) = 0$ and $z \in Z$ then $\zeta'(z) = 1$.
- **Finishing a loop:** Whenever $S'_1 \xrightarrow{\zeta} S'_0 + Z$, there is an edge from (S', S'_0, ζ) to (S'_0, S'_0, ζ'), where $\zeta'(z) = 0$ if $\zeta(z) = 0$ and $z \notin Z$, otherwise $\zeta'(z) = \infty$.

A vertex (S', S', ζ) is designated as "terminal" if $S' + \infty\zeta^{-1}(\infty) \xrightarrow{r}$, that is, if there is some implementation reaction r' with $m(r') = r$ that can occur in S' plus sufficiently many copies of null species z with $\zeta(z) = \infty$.

Some comments on the definition may help give an intuitive understanding of the loopsearch graph. First, note that ζ is monotonic in this graph: for any given $z \in \mathcal{Z}$, $\zeta(z)$ can change from 0 to 1, from 1 to ∞, or from 0 to ∞, but never decrease. A null species z can be produced inside a loop, but the paths we are searching for cannot use z inside the loop where it was first produced; and once that loop ends, z is present in "infinite" copies and will always be so. Second, the loopsearch graph has a repeating substructure that mirrors the structure of the graph of minimal states; compare Fig. 11 bottom right to the remainder of Fig. 11. Vertices of the form (S', S', ζ) for fixed ζ with $\zeta^{-1}(1) = \emptyset$, with edges from "Reactions outside a loop" in Definition 4.2, have exactly the structure of the graph of minimal states, with "greyed" edges (representing reactions that

have null species as reactants, see Fig. 10B, Fig. 11 bottom right) present or absent depending on whether the null species z that are reactants of the corresponding reaction have $\zeta(z) = \infty$. Vertices of the form (S', S'_0, ζ) for fixed S'_0 and ζ, with edges from "Reactions inside a loop" and "Finishing a loop" in Definition 4.2, have a structure very close to the graph of minimal states, differing occasionally when the edge changes ζ. Finally, many of the vertices in the loopsearch graph are unreachable from any "initial vertex" (i.e., vertex of the form $(S', S', \vec{0})$); usually such unreachable vertices, according to the meaning we give them, would contain some sort of contradiction. For example, every vertex of the form (S', S', ζ) with $\zeta^{-1}(1) \neq \emptyset$ will be unreachable in any loopsearch graph; in such a vertex, the form (S', S', ζ) means we are at state S' and not in the middle of a loop, but $\zeta(z) = 1$ means z has been produced in the current, nonexistent loop. Other vertices are unreachable due to less obvious contradictions; in the example in Fig. 11, where $i \in \{1, 2, 3\}$ and $j \in \{4, 5\}$, vertices of the form (S'_i, S'_j, ζ) are unreachable, because we would be at S'_i in a loop starting at S'_j, but such states S'_i are unreachable from states S'_j; similarly, vertices of the form (S'_j, S'_i, ζ) are unreachable for $\zeta(z) \neq \infty$. Unreachable vertices and edges are shown in gray in Fig. 11.

Because the edges in the loopsearch graph come from trivial reactions possible at the corresponding states, any path through the loopsearch graph implies the existence of a class of trajectories in the implementation CRN. A segment from (S'_0, S'_0, ζ) to (S'_1, S'_1, ζ) traveling only through vertices of the form (S', S', ζ) with no change in ζ implies the corresponding sequence of trivial reactions can occur starting from S'_0 plus some null species, including "sufficiently many" copies of $\zeta^{-1}(\infty)$, and ending in a state that covers S'_1. A segment from (S'_0, S'_0, ζ) to (S'_0, S'_0, ζ') traveling only through vertices of the form (S', S'_0, ζ'') implies the existence of a "loop" in the implementation CRN of the form $S'_0 + Z_0 \overset{\tau}{\Rightarrow} S'_0 + Z_1$, where Z_0 includes "sufficiently many" copies of $\zeta^{-1}(\infty)$, and Z_1 includes at least all null species in $(\zeta')^{-1}(\infty)$ that are not in $\zeta^{-1}(\infty)$. Given any number of times for this loop to happen, it can happen that many times starting with sufficiently many copies of $\zeta^{-1}(\infty)$, producing as many copies of $(\zeta')^{-1}(\infty)$ as desired. This logic lets us compose paths made of these segments into trajectories possible in the implementation CRN, by taking each loop as many times as necessary to produce all species necessary for all future segments. In particular, a path from $(S'_0, S'_0, \vec{0})$, where $\vec{0} \in 3^{\mathcal{Z}}$ is the function that maps all null species to 0, to a terminal state of the form (S', S', ζ) implies the existence of a trajectory in the implementation CRN by which $S'_0 \overset{r}{\Rightarrow}$. Thus, if such paths can be found for every minimal S'_0, we know that the permissive condition is satisfied. What we will show is that, if the permissive condition is satisfied, then it is satisfied by trajectories corresponding to paths through the loopsearch graph.

Lemma 4.3. *Let $(\mathcal{S}, \mathcal{R})$ and $(\mathcal{S}', \mathcal{R}')$ be a formal and implementation CRN, with interpretation m. Let $r = (R, P) \in \mathcal{R}$ be a formal reaction and S'_0 an implementation state minimal for $m(S'_0) \geq R$. If the permissive condition is satisfied, then there exists a path through the loopsearch graph described in Definition 4.2 from $(S'_0, S'_0, \vec{0})$ to some terminal state, where $\vec{0}(z) = 0$ for all null species z. Conversely, if such paths exist for* every *formal reaction and minimal implementation state, then the permissive condition is satisfied.*

Proof. Given $r = R \to P \in \mathcal{R}$ and $S'_0 \in \mathcal{M}(r)$, assuming the permissive condition is true, we find a path through the loopsearch graph for r from $(S'_0, S'_0, \vec{0})$ to a terminal state. In particular, we show that if the permissive condition is true, then from any (S'_1, S'_1, ζ) where $\zeta^{-1}(1) = \emptyset$, there is a path either to a terminal state (S'_2, S'_2, ζ) for the same ζ, or to another such state (S'_2, S'_2, ζ') where $(\zeta')^{-1}(1) = \emptyset$ and $\zeta^{-1}(\infty) \subsetneq (\zeta')^{-1}(\infty)$, from which this process can be repeated. Since $\zeta^{-1}(\infty) \subset \mathcal{Z}$ which is finite, this process must find a terminal state in finitely many steps, namely at most $|\mathcal{Z}|$.

Given arbitrary (S'_1, S'_1, ζ) with $\zeta^{-1}(1) = \emptyset$, let $Z = \zeta^{-1}(\infty)$ and note that for each $S'' \in \mathcal{M}(r)$ and $\mu \geq 0$, we know that $m(S'' + \mu Z) \geq R$. By the permissive condition, there is a trajectory in the implementation CRN by which $S'' + \mu Z \overset{r}{\Rightarrow}$; for each S'', choose a shortest such path. Construct a new trajectory by starting at $S'_1 + \mu Z$, where μ is high enough for this trajectory to be valid, and at each step where we are $\geq$ some $S'' + \mu' Z$, take the first reaction on the chosen shortest path for S''. Continue until the trajectory either takes an implementation reaction r' with $m(r') = r$, or covers the same minimal state S'_2 twice.

If the trajectory takes an implementation reaction r' with $m(r') = r$, then by the construction of the loopsearch graph, for each reaction in the trajectory from a state $\geq S''_1$ to a state $\geq S''_2$, there is an edge from (S''_1, S''_1, ζ) to (S''_2, S''_2, ζ). Where S'_2 is the minimal state such that r' was taken in a state $\geq S'_2$, since r' was possible that means (S'_2, S'_2, ζ) is a terminal state. Thus these edges give a path from (S'_1, S'_1, ζ) to (S'_2, S'_2, ζ) which is terminal, which is the desired path.

If on the other hand the trajectory covers the same minimal state S'_2 twice, then there must be at least one null species $z \notin Z$ produced by a reaction between the first and second times S'_2 is covered; otherwise such a path would be a futile loop, implying that for at least one S'' covered in that time there is a μ which gives a shorter path by which $S'' + \mu Z \overset{r}{\Rightarrow}$. Then again by the construction of the loopsearch graph, for each reaction in the trajectory from a state $\geq S''_1$ to a state $\geq S''_2$ *before* the first state $\geq S'_2$, there is an edge from (S''_1, S''_1, ζ) to (S''_2, S''_2, ζ). For each reaction *after* the first state $\geq S'_2$, there is an edge from (S''_1, S'_2, ζ''_1) to (S''_2, S'_2, ζ''_2), with the first $\zeta''_1 = \zeta$, each new ζ''_2 equaling ζ''_1 except that any null species z produced in the corresponding reaction with $\zeta''_1(z) = 0$ has $\zeta''_2(z) = 1$ ("Reaction inside a loop" in Definition 4.2), and the last $\zeta''_2 = \zeta'$ has $\zeta'(z) = \infty$ if any $\zeta''(z) = \infty$ or 1 or if z was produced by the last reaction ($\zeta'(z) = 0$ otherwise), ending in the state (S'_2, S'_2, ζ') ("Finishing a loop"). Since at least one such $z \notin \zeta^{-1}(\infty)$ must have been produced, this is a path from (S'_1, S'_1, ζ) to (S'_2, S'_2, ζ') with $\zeta^{-1}(\infty) \subsetneq (\zeta')^{-1}(\infty)$, which is the desired path.

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

If such a path through the loopsearch graph from $(S'_0, S'_0, \vec{0})$ exists for a given formal reaction r and minimal implementation state S'_0, then we show $S'_0 \overset{r}{\Rightarrow}$. We gave this argument informally above. Where states are of the form (S'_i, S'_j, ζ), observe from Definition 4.2 that the only edges that changes ζ leave S'_j unchanged (i.e. are in a loop), and the only edges that change S'_j are edges from (S'_1, S'_1, ζ) to (S'_2, S'_2, ζ) leaving ζ unchanged with $\zeta^{-1}(1) = \emptyset$ (i.e. are outside a loop). From that, given a path from $(S'_0, S'_0, \vec{0})$ to some terminal state (S'_f, S'_f, ζ_f), we can divide the path into segments as follows: in states of the form (S'_i, S'_j, ζ), segments will alternate between segments where all states have $S'_i = S'_j$ and ζ is unchanged ("paths"), followed by segments where S'_j is unchanged ("loops"). Where (S'_i, S'_i, ζ_i) is the state at the end of the ith loop and $Z_i = \zeta_i^{-1}(\infty)$, we show by induction on i that for any $\mu \geq 0$, $S'_0 \overset{\tau}{\Rightarrow} S'_i + \mu Z_i$. The base case $i = 0$ has $\vec{0}^{-1}(\infty) = \emptyset$, so $S'_0 \overset{\tau}{\Rightarrow} S'_0 + \mu\emptyset$ is trivially true. Assuming that $S'_0 \overset{\tau}{\Rightarrow} S'_i + \mu Z_i$ for any μ, consider the sequence of trivial reactions corresponding to the edges on the path from (S'_i, S'_i, ζ_i) to $(S'_{i+1}, S'_{i+1}, \zeta_i)$ and the loop from there to $(S'_{i+1}, S'_{i+1}, \zeta_{i+1})$. Those trivial reactions are, by Definition 4.2, possible using only null species in Z_i; let μ' be the largest number of a single null species in Z_i used after adding up all reactants of all reactions along the path and loop (ignoring any products of the reactions). Given arbitrary $\mu \geq 0$, let $\mu'' = (\mu + 1)(\mu' + 1)$, and by assumption, $S'_0 \overset{\tau}{\Rightarrow} S'_i + \mu'' Z_i$. Then by following the trivial reactions along the path from S'_i to S'_{i+1}, we use up at most $\mu' Z_i$, so $S'_i + \mu'' Z_i \overset{\tau}{\Rightarrow} S'_{i+1} + \mu(\mu' + 1)Z_i$. Then by following the trivial reactions along the loop μ times, each loop uses at most $\mu' Z_i$ and produces at least $Z_{i+1} \backslash Z_i$, so $S'_{i+1} + \mu(\mu' + 1)Z_i \overset{\tau}{\Rightarrow} S'_{i+1} + \mu Z_i + \mu(Z_{i+1} \backslash Z_i)$, completing the induction. That $S'_0 \overset{\tau}{\Rightarrow} S'_f + \mu Z_f$ for all $\mu \geq 0$ is a special case of this proof, and since by Definition 4.2 that (S'_f, S'_f, ζ_f) is a terminal state means $S'_f + \mu Z_f \overset{r}{\rightarrow}$ for some $\mu \geq 0$, this proves that $S'_0 \overset{r}{\Rightarrow}$.

If such paths exist for every formal reaction r and minimal state S'_0 for r, then every minimal state $S'_0 \overset{r}{\Rightarrow}$. Thus as discussed in Lemma 4.1 every state with $m(S') \overset{r}{\rightarrow}$ has $S' \overset{r}{\Rightarrow}$, satisfying the permissive condition. □

With this preparation, we can now describe algorithms to check the permissive condition. Having shown that the permissive condition is true if and only if certain paths through the loopsearch graph exist, our algorithms will be based on searching for those paths. In general, if a formal reaction $r = R \rightarrow P$ has $k = |R|$ reactants and the implementation CRN has $n = |\mathcal{S}'|$ species, there may be order n^k minimal implementation states for r and the trajectories by which any one implements r may have to pass through most or all of them. As that suggests, we will later prove that checking the permissive condition (and thus checking an interpretation in general) is PSPACE-complete. So the first algorithm we present is the loopsearch algorithm, which runs in poly(n, k) space and poly(n^{kn}) time, which is Algorithm 1.

The size (number of vertices) of the loopsearch graph is $|\mathcal{M}(r)|^2 3^{|\mathcal{Z}|}$, at worst exponential in the size of the CRNs, and we have reduced the permissive condition to a question of whether paths between certain pairs of vertices exist in that graph. Savitch's theorem states that we can decide whether such paths exist through a graph of size N in $\log^2 N$ space [36], which given the results so far completes the proof that the permissive condition can be decided in polynomial space; the loopsearch algorithm is just a concrete application of Savitch's result to the loopsearch graph. Specifically, the loopsearch algorithm breaks a path from $(S'_0, S'_0, \vec{0})$ into alternating segments of two types: one type from (S'_i, S'_i, ζ_i) to $(S'_{i+1}, S'_{i+1}, \zeta_i)$ through only states of the form (S', S', ζ_i), and the other type from $(S'_i, S'_i, \zeta_{i-1})$ to (S'_i, S'_i, ζ_i) through only states of the form (S', S'_i, ζ), the same decomposition discussed in the proof of Lemma 4.3. (In the preliminary version of this paper we called such decomposition a *loop-segmented path* [32].) Both types of segments can have length no longer than $|\mathcal{M}(r)|$; for the first type, this is obvious, while for the second, we rely on the proof of Lemma 4.3 to say that a decomposition exists where no segment covers the same minimal state twice. To search for a path of length $2^0 = 1$, we check each possible edge (trivial reaction) to see if the start and target state are connected; to search for a path of length 2^{i+1}, we check for each possible intermediate state, whether a path of length 2^i exists from the start to the intermediate, and whether a path of length 2^i exists from the intermediate to the target. For a segment of the first type, the possible intermediate states are just (S', S', ζ_i) for $S' \in \mathcal{M}(r)$, while for a segment of the second type, the possible intermediate states are (S', S'_i, ζ) where $S' \in \mathcal{M}(r)$ and for all $z \in \mathcal{Z}$, $\zeta_{i-1}(z) \leq \zeta(z) \leq \zeta_i(z)$. This condition, which reduces the number of ζ to check, relies on a monotonicity of $\zeta(z)$ that follows from the types of edges defined in Definition 4.2.

Theorem 4.1. *Whether an interpretation is a bisimulation can be checked in polynomial space.*

Proof. We show that the loopsearch algorithm is correct and runs in polynomial space. We proved in Lemma 4.3 that the permissive condition is true if and only if a loop-segmented path exists for every formal reaction r and every minimal implementation state S'_0 for r, so we need only to show that the loopsearch algorithm finds a loop-segmented path if and only if one exists. Each loop-segmented path implicitly specifies a sequence of l minimal states S'_i, and a sequence of sets of null species Y_i. By removing from Y_i all null species in Y_j for $j < i$ and defining $Y_0 = Z - \bigcup_{i=1}^{l} Y_i$, we get the partition that the loopsearch algorithm searches for while preserving the loop-segmented path. Since the loops and paths in the desired loop-segmented path never repeat a minimal state, they must each have length at most N the number of minimal states, and a path of length 2^j from S'_a to S'_b exists if and only if for some S'_c a path of length 2^{j-1} from S'_a to S'_c and a path of

Algorithm 1: The loopsearch algorithm to check the permissive condition in polynomial space.

```
def loopsearch(CRN formal, CRN impl, interpretation m):
  Z = { species x in impl where m(x) is empty }
  for each reaction r in formal:
    Min = minimal_states(impl, m, r)
    k = log(|Min|, base 2)
    for each state S'0 in Min:
      found = False
      for each sets <Y0,Y1,...,Yl> in partitions(Z,|Z|+1), states <S'1,...,S'l> in Min:
        if (for all i in 1,...,l, ...
            reach_with_inf(S'(i-1), S'i, Y1+...+Y(i-1), k) ...
            and reach_with_inf(S'i, S'i + Yi, Y1+...+Y(i-1), k)) ...
           and reach_with_inf(S'l, r, Y1+...+Yl, k):
          found = True
      if not found:
        return False

  # if found for all r and all S'0 ...
  return True

def reach_with_inf(state start, (state or reaction) target, ...
                   (set of species) infinites, integer k):
  # check if start can reach target in at most 2^k reactions
  #  given infinitely many copies of species in infinites
  if k is 0:
    for each trivial reaction r':
      if r' takes start to target: return True
    return False
  if k is not 0:
    middles = Min x 2^({z in target | m(z) = 0})
    for each state middle in middles:
      if reach_with_inf(start, middle, infinites, k-1) ...
          and reach_with_inf(middle, target, infinites, k-1):
        return True
    return False
```

length 2^{j-1} from S'_c to S'_b both exist. The desired loop-segmented path has each loop and path as a path between minimal states with certain null species ignored and the algorithm matches this restriction, so the loopsearch algorithm is correct.

At any point in the loopsearch algorithm, it is storing the following information: a formal reaction r, a minimal state S'_0, a partition of $l+1$ sets of null species Y_i (thus implying that $l \leq z \leq n$), a sequence of l minimal states S'_i, and at most $\lceil \log N \rceil$ triples of minimal states S'_a, S'_b, S'_c in the recursive search algorithm. The at most n^k minimal states can be enumerated in polynomial space (i.e. without storing any states other than the current and next one) as shown in Lemma 4.2, and similarly partitions of $z \leq n$ elements can be enumerated in poly(z) space. Also according to Lemma 4.2, the number of minimal states is $N \leq n^k$, so the depth of the search $\lceil \log N \rceil$ is poly(n, k). (When $k \geq n$, $N \leq 2^{n \log k}$ so the depth is at most poly$(n, \log k)$.) To complete the proof, note that as discussed earlier, checking the atomic and delimiting conditions are both trivial given an interpretation, thus whether an interpretation is a bisimulation can be checked in polynomial space. □

We have repeatedly said that the difficulty of checking the permissive condition scales with the number of minimal states for any given formal reaction $r = R \to P$, which typically scales like (and scales no worse than) n^k where $n = |\mathcal{S}'|$ and $k = |R|$. We stated earlier, and will show later, that when k is unbounded, checking an interpretation is PSPACE-complete. However, many CRNs in practice have large numbers of species but small numbers of reactants per (formal) reaction; in particular, almost any interesting behavior—if not any interesting behavior—that can be done with a CRN can be done with a CRN with a bound of $k \leq 2$. For those CRNs, we present a graphsearch algorithm which takes poly(n^k) space and time, making it much faster than the loopsearch algorithm when k is small but taking much more space when k is large, as Algorithm 2.

For each formal reaction r, the graphsearch algorithm enumerates and creates a table of all implementation states S'_i minimal for r. The algorithm uses this table to store "known information" about which states are reachable from S'_i and iteratively updates this information, continuing until either every $S'_i \overset{r}{\Rightarrow}$ is known or until no further information can be known, in which case some $S'_i \overset{r}{\not\Rightarrow}$. For each S'_i, the algorithm stores whether or not it is known (yet) that $S'_i \overset{r}{\Rightarrow}$. If it is not yet known that $S'_i \overset{r}{\Rightarrow}$, then the algorithm stores, for each minimal state S'_i, whether it is known that $S'_i \overset{\tau}{\Rightarrow} S'_j$, and for each $y \in \mathcal{S}'$ with $m(y) = \emptyset$, whether it is known that $S'_i \overset{\tau}{\Rightarrow} S'_i + y$. Initially, the only thing known is that $S'_i \overset{\tau}{\Rightarrow} S'_i$ for each S'_i.

Given this table, the algorithm goes through repeated "cycles" of updating the known reachabilities until one cycle passes with no changes. In each cycle, the algorithm iterates through each minimal S'_i. For each S'_i, if $S'_i \overset{r}{\Rightarrow}$ is known, the algorithm

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

Algorithm 2: The graphsearch algorithm to check the permissive condition in time and space polynomial in the number of minimal states.

```
def graphsearch(CRN formal, CRN impl, interpretation m):
  for each reaction r in formal:
    Min = minimal_states(impl, m, r)
    # states or r reachable from S'
    table reach(S') = <empty> for each state S' in Min
    # null species producible in a loop at S'
    table prod(S') = <empty> for each state S' in Min
    repeat until reach and prod are both unchanged:
      for each S'1 where reach(S'1) is not r:
        if S'1 + inf prod(S'1) can do r' with m(r') =  r:
          set reach(S'1) to r, continue to next S'1
        for each trivial reaction r' by which S'1 + inf prod(S'1) -> S'2:
          if reach(S'2) is r:
            set reach(S'1) to r, continue to next S'1
          reach(S'1) += reach(S'2)
          if S'1 in reach(S'2):
            prod(S'1) += prod(S'2)
            prod(S'1) += {y in products(r') where m(y) = empty}
    if not all reach(S') is r: # after table no longer changes
      return False
  return True # if all reactions pass without returning False
```

skips S'_i. If not, where Y_i is the set of all y such that $S'_i \stackrel{\tau}{\Rightarrow} S'_i + y$ is known, the algorithm checks for each implementation reaction r' with $m(r') = r$ whether it can occur in S'_i given arbitrarily many copies of Y_i. If $S'_i + \infty Y_i \xrightarrow{r'}$ for one of those r', then the algorithm records that $S'_i \stackrel{r}{\Rightarrow}$ and will skip S'_i in the future. Otherwise, the algorithm iterates through all trivial reactions in the implementation CRN and check whether they can occur in $S'_i + \infty Y_i$. For each reaction that can occur, the algorithm finds the (possibly multiple) minimal state(s) S'_j that are covered by the state after that reaction, and updates the table based on what is known about S'_j:

(i) If $S'_i + \infty Y_i \xrightarrow{\tau} S'_j$ and $S'_j \stackrel{r}{\Rightarrow}$, then $S'_i \stackrel{r}{\Rightarrow}$. Otherwise,
(ii) If $S'_i + \infty Y_i \xrightarrow{\tau} S'_j$ and $S'_j \stackrel{\tau}{\Rightarrow} S'_k$, then $S'_i \stackrel{\tau}{\Rightarrow} S'_k$.
(iii) If $S'_i + \infty Y_i \xrightarrow{\tau} S'_j + y$ and $S'_j \stackrel{\tau}{\Rightarrow} S'_i$, then $S'_i \stackrel{\tau}{\Rightarrow} S'_i + y$.
(iv) If $S'_i + \infty Y_i \xrightarrow{\tau} S'_j$ and $S'_j \stackrel{\tau}{\Rightarrow} S'_j + y$ and $S'_j \stackrel{\tau}{\Rightarrow} S'_i$, then $S'_i \stackrel{\tau}{\Rightarrow} S'_i + y$.

The algorithm terminates when a full cycle passes with no change to the table. At that time, if $S'_i \stackrel{r}{\Rightarrow}$ is known for every minimal S'_i, the algorithm states that the permissive condition is true; otherwise, the algorithm states that the permissive condition is false.

Theorem 4.2. *When the number of reactants in a formal reaction k is constant, whether an interpretation is a bisimulation can be checked in polynomial time.*

Proof. We prove that i) if the graphsearch algorithm returns true, then the permissive condition is true; ii) if the permissive condition is true, then the graphsearch algorithm will return true; and iii) the graphsearch algorithm always terminates in poly(n^k) time.

To prove the first, the graphsearch algorithm is based on deductions from "known" information. The initially known information is that each $S'_i \stackrel{\tau}{\Rightarrow} S'_i$, which is trivially true (since $\stackrel{\tau}{\Rightarrow}$ includes the sequence of 0 reactions). There are five deduction rules: the rule that if $S'_i \stackrel{\tau}{\Rightarrow} S'_i + Y_i$ and $S'_i + \infty Y_i \xrightarrow{r'}$ with $m(r') = r$ then $S'_i \stackrel{r}{\Rightarrow}$, and the four listed rules for when $S'_i + \infty Y_i \xrightarrow{\tau} S'_j$. Each of the rules is a valid deduction, so any information deduced by the algorithm will be true. In particular, the algorithm says the permissive condition is true only when every minimal $S'_i \stackrel{r}{\Rightarrow}$, which by Lemma 4.1 implies that the permissive condition is in fact true.

To prove the second, we show that if the permissive condition is true but at a given cycle evaluating the formal reaction r the graphsearch algorithm does not yet know that every minimal $S'_i \stackrel{r}{\Rightarrow}$, then there is at least one additional fact that it can learn this cycle. The proof is similar, but not identical, to the proof of Lemma 4.3. At any given cycle, for each minimal state S'_i let Y_i be the set of all null species such that $S'_i \stackrel{\tau}{\Rightarrow} S'_i + Y_i$ is known. If the permissive condition is true, then the permissive condition is true in a modified CRN which for each S'_i adds the (trivial under the given interpretation) reaction

$S'_i \to S'_i + Y_i$. For each minimal S'_i, there is a path in the modified CRN by which $S'_i \overset{r}{\Rightarrow}$ which is "shortest" in the sense of having the fewest reactions that are *not* any of the added reactions $S'_i \to S'_i + Y_i$; consider for each of those shortest paths the first reaction that is not one of the added reactions. Note that each of those reactions is a reaction, either trivial or implementing r, that the graphsearch algorithm will detect as possible and consider.

For each S'_i, construct a trajectory which starts at S'_i, takes the reaction $S'_i \to S'_i + Y_i$ as many times as necessary to take the first non-added reaction on the shortest path by which $S'_i \overset{r}{\Rightarrow}$, and takes that reaction to reach some S'_j; takes $S'_j \to S'_j + Y_j$ as many times as possible to take the first non-added reaction on the shortest path by which $S'_j \overset{r}{\Rightarrow}$, and takes that reaction; then continues this process. Each such path must eventually, in a number of reactions less than the number of minimal states $N \leq n^k$, either implement r or repeat a minimal state. If any path eventually implements r, and for any minimal state S'_k on that path it is not known that $S'_k \overset{r}{\Rightarrow}$, then the last such S'_k has the reaction $S'_k \overset{\tau}{\to} S'_l$ available and $S'_l \overset{r}{\Rightarrow}$ known, so on this cycle the algorithm will deduce that $S'_k \overset{r}{\Rightarrow}$.

The last possibility is that at least one path much eventually repeat a minimal state; if all paths implement r and all states on such paths are known to implement r, then all minimal states are known to implement r. For any such loop, that loop will be the entire trajectory for all states on that loop. If some state in that loop is not known to reach some other state in that loop, then at least one such fact will be deduced this cycle: there will be some S'_i where a reaction $S'_i \overset{\tau}{\to} S'_j$ is possible and $S'_j \overset{\tau}{\Rightarrow} S'_k$ is known but $S'_i \overset{\tau}{\Rightarrow} S'_k$ is not known; that fact will be deduced this cycle. If not, then there must be some y with $m(y) = \emptyset$ that is produced along that loop and some S'_i in that loop for which $S'_i \overset{\tau}{\Rightarrow} S'_i + y$ is not known; otherwise one of the reactions in the loop is not the first reaction along the shortest path by which the appropriate $S'_j \overset{r}{\Rightarrow}$. If that S'_i is one such that $S'_i \overset{\tau}{\to} S'_j + y$, then $S'_j \overset{\tau}{\Rightarrow} S'_i$ is known (by assumption that all such facts in this loop are known), and $S'_i \overset{\tau}{\Rightarrow} S'_i + y$ will be deduced this cycle. Otherwise, there will be an S'_i such that $S'_i \overset{\tau}{\to} S'_j$ is possible, $S'_j \overset{\tau}{\Rightarrow} S'_j + y$ is known but $S'_i \overset{\tau}{\Rightarrow} S'_i + y$ is not known, and that fact will be deduced this cycle. This covers all the cases, and proves that if the permissive condition is true but not yet proven, then at least one fact will be deduced each cycle until the permissive condition is proven.

To complete the proof, we note that the number of facts is bounded above by $(z+1)n^k + n^{2k}$, where $z \leq n$ is the number of null species, and thus is poly(n^k). Thus, if the permissive condition is true, one of a finite number of facts will be learned each cycle until the permissive condition is proven. Furthermore, the algorithm will terminate one way or another in at most poly(n^k) cycles, thus poly(n^k) time. □

Although polynomial space is inefficient, in the general case we cannot do better. Two results in particular suggest a connection between CRNs and space-bounded Turing machines, the acceptance problem of which is known to be PSPACE-complete [37]; we use this connection to prove that verifying CRN bisimulation is PSPACE-complete. Jones et al. gave a construction to, given a space-bounded Turing machine with m states and tape size n, construct a Petri net (equivalently, a CRN) that directly simulates it, with poly(n, m) species and reactions [38]. Thachuk and Condon extended this connection to reversible CRNs, constructing a CRN that solves the known PSPACE-complete problem QSAT, proving a number of questions about CRNs and DNA strand displacement systems to be PSPACE-complete [39]. In the case of CRN bisimulation, if we have (on the order of) n^k minimal states, it is possible to embed a PSPACE-complete computation in the trivial reactions between those n^k states. Given any space-bounded Turing machine and input, we construct a formal and implementation CRN with interpretation, where the implementation CRN contains the construction of Jones et al. in the trivial reactions, plus some additional reactions specific to our case. Our construction is illustrated in Fig. 12. There is one formal reaction that can only occur in an implementation state corresponding to the accept state of the Turing machine; thus, the state corresponding to the start state can implement that reaction if and only if the Turing machine does in fact accept. The additional reactions ensure that every minimal implementation state can implement the formal reaction if the "start state" can, making the interpretation a CRN bisimulation if and only if the space-bounded Turing machine accepts.

Theorem 4.3. *Verifying CRN bisimulation in the general case is PSPACE-complete.*

Proof. We are given a Turing machine with m states with tape alphabet $\{0, 1\}$, and an input x of length n. We assume the states are numbered such that q^0 is the start state and q^{m-1} is the halt state. We assume the Turing machine always halts while never using more space than the length of its input, and when it halts, it does so in state q^{m-1} reading the first square of its tape, with all squares but the first reading 0, and the first square reading 1 to indicate an accepting state and 0 to indicate rejecting. Given that, our formal CRN has $n+2$ species and 1 reaction, $Q + A_1 + \cdots + A_n \to H$. We construct an implementation CRN with species 0_i and 1_i for each spot on the tape $1 \leq i \leq n$, q_i^j for each tape spot $1 \leq i \leq n$ and state of the Turing machine $0 \leq j \leq m-1$, additional species for a "reset" state q_i^m for $1 \leq i \leq n$, and a halting species h. The implementation CRN contains reactions to simulate the Turing machine, reactions to reset the Turing machine to the start state q^0 on input x, and reactions to check whether a halting state is accepting or rejecting that can implement the formal reaction if and only if it is an accepting state.

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

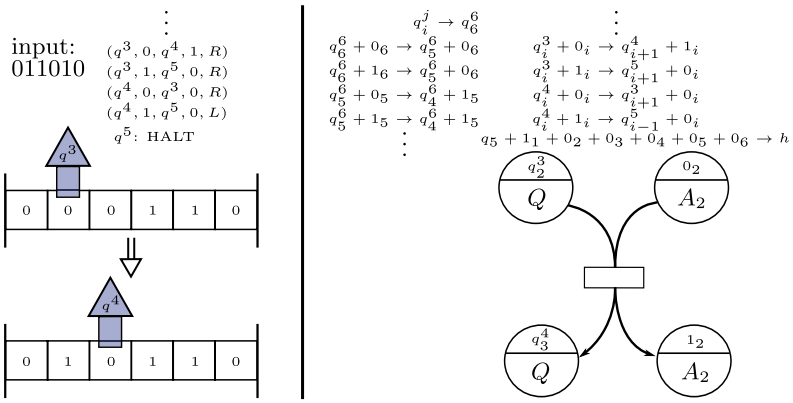


Fig. 12. An implementation CRN with a correct interpretation if and only if the corresponding space-bounded Turing machine accepts. The formal CRN has one species Q corresponding to the Turing machine head and one species A_i for each ith tape square; if all are present, they can react. The implementation CRN simulates the space-bounded Turing machine, with Turing machine head state species q_i^j all interpreted as Q and tape square species 0_i and 1_i both interpreted as A_i. All reactions involved in the simulation are thus trivial. If the simulation accepts, the formal reaction can be implemented. At any time the implementation CRN can use q_i^6 to "reset" to the start state on the given input, thus being able to "correctly" simulate the computation from an arbitrary initial implementation state. Thus this interpretation satisfies the permissive condition if and only if the Turing machine accepts.

To simulate the Turing machine, for each transition of the form, in state j reading symbol $\sigma \in \{0, 1\}$, write symbol $\sigma' \in \{0, 1\}$, transition to state j', and move (right, left) we have n reactions of the form $q_i^j + \sigma_i \to q_{i\pm1}^{j'} + \sigma_i'$ for $1 \leq i \leq n$, where the product is $q_{i+1}^{j'}$ if the move is right and $q_{i-1}^{j'}$ if the move is left. (If the move is right, the reaction for q_n^j is instead $q_n^j + \sigma_n \to q_n^{j'} + \sigma_n'$, and similarly if the move is left the reaction for q_1^j is instead $q_1^j + \sigma_1 \to q_1^{j'} + \sigma_1'$.)

To reset, we have a reaction $q_i^j \to q_n^m$ for every q_i^j including $j = m$ and q_1^{m-1} but not including any q_i^{m-1} for $i \geq 2$. We then have reactions $q_i^m + \sigma_i \to q_{i-1}^m + x_i$ for each $2 \leq i \leq n$ and $q_1^m + \sigma_1 \to q_1^0 + x_1$, in each case for both $\sigma = 0$ and $\sigma = 1$, where x_i represents the species 0_i if the ith character of the string x is 0 and 1_i if the ith character is 1.

To check whether a halting state is an accepting state, we have a reaction $q_1^{m-1} + 1_i \rightleftharpoons q_2^{m-1}$, reactions $q_i^{m-1} + 0_i \rightleftharpoons q_{i+1}^{m-1}$ for $2 \leq i \leq n-1$, and a reaction $q_n^{m-1} + 0_n \to h$. Note that by assumption q^{m-1} is the halting state of the given Turing machine and thus has no transitions; so other than these reactions the only implementation reaction with any q_i^{m-1} as a reactant is the reaction $q_1^{m-1} \to q_n^m$.

We want to check the validity of the interpretation m where $m(q_i^j) = Q$ for $j \neq m-1$, $m(q_i^{m-1}) = Q + \sum_{k=1}^{i-1} A_k$ (so for example $m(q_1^{m-1}) = Q$ and $m(q_3^{m-1}) = Q + A_1 + A_2$), $m(0_i) = m(1_i) = A_i$, and $m(h) = H$. However, we will show in Theorem 4.5 that for any CRN constructed this way based on a Turing machine, the only possible valid interpretations are this one up to a permutation of formal species, and either this interpretation is valid or no valid interpretation exists.

We use the three conditions formulation of validity. The atomic condition is always satisfied by $m(q_1^0) = Q$, $m(0_i) = A_i$, and $m(h) = H$, as well as many other ways. The delimiting condition is satisfied since under this interpretation, every reaction mentioned above is trivial except for the reaction $q_n^{m-1} + 0_n \to h$, which is interpreted as the one formal reaction $Q + A_1 + \cdots + A_n \to H$. It only remains to check the permissive condition, and we will show that the permissive condition is true if and only if the given Turing machine accepts the given input x.

The set of minimal states for the one formal reaction $r = Q + A_1 + \cdots + A_n \to H$ is exactly the set of states containing either one copy of one q_i^j for $j \neq m-1$ and one copy of either 0_i or 1_i for each $1 \leq i \leq n$, or one copy of some q_i^{m-1} and one copy of either 0_k or 1_k for each $i \leq k \leq n$. Appealing to Lemma 4.1, the permissive condition is true if and only if each of those minimal states S' has $S' \overset{r}{\Rightarrow}$. We are particularly interested in the state S_0' containing q_1^0 and the species x_i for each $1 \leq i \leq n$, where x_i represents either 0_i or 1_i depending on the ith character of x, which is minimal. Any minimal state of the first type with a q_i^j for $j \neq m-1$ can reach S_0' by the reaction $q_i^j \to q_n^m$ followed by the reset reactions $q_i^m + \sigma_i \to q_{i-1}^m + x_i$ and $q_1^m + \sigma_1 \to q_1^0 + x_1$ in the appropriate order. Any minimal state of the second type with a q_i^{m-1} can reach S_0' by the reverse reactions $q_{i+1}^{m-1} \to q_i^{m-1} + 0_i$ and $q_2^{m-1} \to q_1^{m-1} + 1_1$, as appropriate, of the reactions to check the halting state, followed by $q_1^{m-1} \to q_n^m$, followed by the rest of the reset reactions as appropriate. Since every minimal state can reach S_0' via trivial reactions, the permissive condition is true if and only if $S_0' \overset{r}{\Rightarrow}$.

Note that any state of the implementation CRN with exactly one copy of one q_i^j for $j < m-1$ and for each $1 \leq i \leq n$ exactly one of either 0_i or 1_i corresponds to the Turing machine state where the tape contents of square i is whichever of 0_i or 1_i is present, and the Turing machine is in state j reading square i where q_i^j is the q species present. S_0' is one such state. Note also that in any such state, the only possible reactions are to either faithfully simulate the next transition of the Turing machine, leading to either another such state or a halting state (i.e. a state which would be a simulating state

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

except that the q species present is q_1^{m-1}), or to start a reset. In a "resetting" state, which is a state where any q_i^m and for each $1 \leq i \leq n$ exactly one of either 0_i or 1_i is present, the only possible reaction is to continue the reset, leading to either another resetting state or eventually to a state that represents a Turing machine state.

If the Turing machine accepts x, then S_0' can faithfully simulate the Turing machine until it halts, which since the input was x, will be the accepting state $q_1^{m-1} + 1_1 + 0_2 + \cdots + 0_n$. From this state, the reactions to check whether a halting state is an accepting state are possible in order, eventually leading to the reaction $q_n^{m-1} + 0_n \to h$, which is interpreted as r; thus $S_0' \overset{r}{\Rightarrow}$ and the permissive condition is true. Conversely, if the Turing machine rejects x, then from S_0' the only possible trajectories are those that faithfully simulate the Turing machine with occasional resets. Some of those trajectories may reach a halting state, but since the Turing machine rejects x, that state will be $q_1^{m-1} + 0_1 + 0_2 + \cdots + 0_n$, and the reaction $q_1^{m-1} + 1_1 \to q_2^{m-1}$ will not be possible. None of these trajectories ever reach the reaction $q_n^{m-1} + 0_n \to h$, thus $S_0' \overset{r}{\not\Rightarrow}$ and the permissive condition is false. □

4.2. Finding an interpretation

We now consider the problem of, given a formal and implementation CRN, can we find an interpretation that is a bisimulation or correctly assert that none exists? It is natural to consider performing an exhaustive depth-first search through the space of possible interpretations, testing each one to see if it satisfies the atomic, delimiting, and permissive conditions using the algorithms described above—thus either finding an interpretation or asserting that none exists. There are two major stumbling blocks to this approach. First, the space of possible interpretations is infinite, and thus we need some way to guarantee that if a valid interpretation exists, there must be one among a defined finite subset of interpretations that we can search. Second, to be useful in practice, the depth-first search must prune aggressively to eliminate fruitless branches.

The reactionsearch algorithm, presented below, addresses both of these challenges. Rather than directly exploring the space of interpretations, the reactionsearch algorithm organizes the depth-first search according to properties that the interpretation must have, effectively proceeding in five stages. First, as a precondition for the permissive condition, the algorithm ensures that every formal reaction has an implementation reaction that interprets to it; second, to satisfy the delimiting condition, the algorithm ensures that every remaining implementation reaction is interpreted as some formal reaction or is trivial; third, to satisfy the atomic condition, the algorithm ensures that every formal species has some implementation species that interprets to it; fourth, any unassigned implementation species are provided an interpretation that respects the assignment of implementation reactions as formal reactions or trivial reactions; and fifth, the permissive condition is tested on any such completed interpretation that is thus found. We will first describe the algorithm itself, and then discuss the lemmas that guarantee that a valid interpretation will be found if one exists.

Often, implementation CRNs are designed with specific interpretations in mind for some species, so it is reasonable to provide such information as an additional constraint on the search. Further, such a formulation enables a natural recursive definition for the algorithm (Algorithm 3). Thus, the algorithm takes as input a formal CRN $(\mathcal{S}, \mathcal{R})$, implementation CRN $(\mathcal{S}', \mathcal{R}')$, and a partial interpretation m which, for some (possibly empty) subset of $\mathcal{S}'$, specifies each $m(x) \in \mathbb{N}^{\mathcal{S}}$. The algorithm first constructs a table of, for each formal reaction $r \in \mathcal{R}$ or τ and for each implementation reaction $r' \in \mathcal{R}'$, whether r' can be interpreted as r (in some completion of the partial interpretation m, regardless of what that completion will do to the other reactions). The algorithm then enumerates interpretations by iterating, in an order described below, through all possible assignments of each r' to be interpreted as some r or τ, and enumerating completed interpretations which match that.

After constructing the table, if there is some $r \in \mathcal{R}$ with no r' that *is* interpreted as r (i.e., all species x involved in r' have $m(x)$ specified and $m(r') = r$), the algorithm chooses such an r with the smallest number of r' that *can be* interpreted as that r. (If there is an r with no r' that *can be* interpreted as r, then there is no completion of the partial interpretation that can satisfy the atomic and permissive conditions, so the algorithm returns that fact.) For each r' that can be interpreted as that r, the algorithm enumerates all possible interpretations of each species not yet interpreted by m and involved in r' that make $m(r') = r$. For each enumerated set of interpretations, the algorithm calls itself recursively to enumerate completions of the partial interpretation m with those new interpretations added. If a valid completion is found, the algorithm returns it; otherwise, the algorithm continues with the next partial interpretation or next r'.

If after constructing the table every $r \in \mathcal{R}$ has some r' that *is* interpreted as r, the algorithm then finds the r' that has the fewest $r \in \mathcal{R}$ such that r' can be interpreted as r, and which hasn't yet been used for branching. (Again, if there is an r' with no $r \in \mathcal{R}$ *or* τ that r' can be interpreted as, then no completion of the partial interpretation can satisfy the delimiting condition, and the algorithm returns that.) For each such $r \in \mathcal{R}$, the algorithm as above enumerates all possible interpretations of each uninterpreted species involved in r' that make $m(r') = r$ and calls this algorithm recursively for each such partial interpretation. If r' can be interpreted as τ, then an additional recursive branch is explored wherein $m(r') = \tau$ is enforced.

If all $r' \in \mathcal{R}'$ which involve uninterpreted species can be interpreted as τ, then on one branch the algorithm will consider the possibility that all of them are interpreted as τ, which as described above does not involve specifying any interpretations for the remaining uninterpreted species. Since the trivial reaction solver—which as described below will complete the interpretation for the uninterpreted species, if possible—works more efficiently with a partial interpretation which satisfies the atomic condition, the algorithm first ensures that that is the case. For each formal species A for which there is *no* im-

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

Algorithm 3: The reactionsearch algorithm to complete a partial interpretation or assert that no completion exists, in polynomial space.

```
def reactionsearch(CRN formal, CRN impl, partial interpretation m):
  return complete(formal, impl, m, { })
def complete(CRN formal, CRN impl, partial interpretation m, assigned reactions k):
  table maybe(r, r') = True if m does not rule out m(r') = r ...
        for reaction r in formal or trivial, reaction r' in impl
  if any r has no r' or any r' has no r where maybe(r,r'):
    return False
  let r in formal where no r' in impl where m(r') = r ...
      and which minimizes |{r' in impl where maybe(r,r')}|
  if such an r exists:
    for each r' where maybe(r,r'):
      for each assignment of m'(x) where x in r' ...
          and m(x) undefined such that (m U m')(r') = r:
        out = complete(formal, impl, m U m', k U {r'})
        if out is an interpretation: return out
    return False # if no r' is found
  if no such r exists:
    if (m(r') is known or maybe(trivial, r')) for all r':
      for each assignment of,
            for each formal species A with no implementation species x with m(x) = A,
            one unassigned x to have m'(x) = A:
        out = solve_diophantine_equations(impl, m U m')
        if out is an interpretation and permissive_check(out): return out
    if k includes all implementation reactions: return False
    let r' in impl where r' is not in k ...
        and which minimizes |{r in formal where maybe(r,r')}|
    for each r in formal or trivial where maybe(r,r'):
      for each assignment of m'(x) where x in r' ...
          and m(x) undefined such that (m U m')(r') = r:
        out = complete(formal, impl, m U m', k U {r'})
        if out is an interpretation: return out
    return False # if no r is found
```

plementation species x_A where the partial interpretation specifies $m(x_A) = A$, the algorithm lists all possible x_A for which, if $m(x_A) = A$ was added to the partial interpretation, all remaining reactions would still be able to be trivial. The algorithm then iterates over all combinations of choices of such x_A for each unimplemented formal species A, and runs the trivial reaction solver for each combination.

To find a completed interpretation in which all remaining r' are interpreted as τ, the trivial reaction solver sets up and solves a system of linear equations in variables $m(x; A)$ for each uninterpreted implementation species x and formal species A, where $m(x; A)$ is the count of A in $m(x)$. For each pair of an implementation reaction r' and a formal species A the algorithm derives one equation regarding various $m(x; A)$ by setting the sum of counts $m(x; A)$ in the (interpreted or uninterpreted) reactants of r' minus that of the products of r' to be 0. For example, if the implementation reaction $x_1 + x_2 \to x_3 + x_4$ should be interpreted as τ, $m(x_1) = A + C$ and $m(x_3) = B + C$ are specified while $m(x_2)$ and $m(x_4)$ are unspecified, then the algorithm will derive the equations $1 + m(x_2; A) - m(x_4; A) = 0$, $m(x_2; B) - 1 - m(x_4; B) = 0$, $m(x_2; C) - m(x_4; C) = 0$, and $m(x_2; D) - m(x_4; D) = 0$ for each other formal species D. Combining such equations for all remaining implementation reactions that are to be interpreted as trivial, we obtain a system of linear Diophantine equations where the variables specify the interpretations of all remaining uninterpreted species. The trivial reaction solver then runs an algorithm described by Contejean and Devie [40] that will find a minimal solution to a system of linear Diophantine equations, if any solution exists; this solution is then used as the interpretation of each remaining implementation species. (A minimal solution for a system of linear Diophantine equations is one such that no other solution has *every* variable being less than or equal to the given solution; thus there may be many minimal solutions.) We will prove in Lemma 4.4 that if there is any solution to these equations that satisfies the permissive condition, then the minimal solution returned by this algorithm does. If no solution to the equations exists, then no completed interpretation where all remaining $m(r') = \tau$ is possible, and the algorithm returns that.

Once the reactionsearch algorithm has a completed interpretation, which may be passed in initially, passed in by a recursive call, or found by the trivial reaction solver, it then runs either the loopsearch algorithm or the graphsearch algorithm as described previously, or any other algorithm yet to be discovered, in order to check the permissive condition. If the permissive condition is satisfied, then the given interpretation is valid and the reactionsearch algorithm returns that. If not, the algorithm passes that information to previous recursive calls in order to check further possible interpretations. If any level of recursion in the algorithm has checked all possible completed interpretations without finding a valid one, that level returns that no completion exists. An example of the depth-first search tree explored by the reactionsearch algorithm for a simple pair of CRNs is shown in Fig. 13.

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

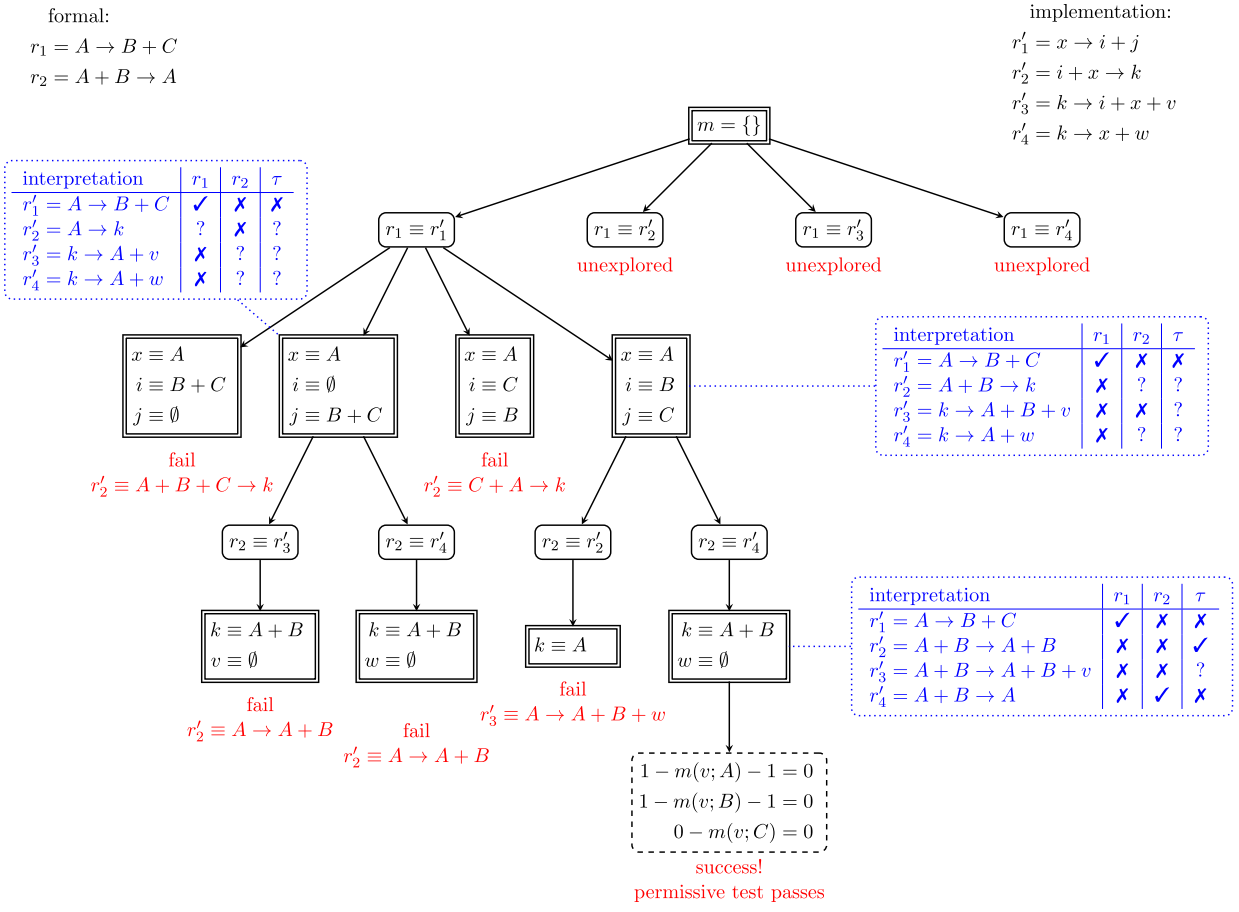


Fig. 13. A pictorial illustration of the search tree explored by the reactionsearch algorithm for the given pair of formal and implementation CRNs. Double-lined boxes indicate the new constraints on the partial interpretation at each node of the tree, where $x \equiv A$ is shorthand for $m(x) = A$. Rounded boxes indicate the new constraints on the interpretation of reactions, where $r \equiv r'$ is shorthand for requiring that $m(r') = r$. The dashed box indicates the Diophantine equation set up by the trivial reaction solver upon the first execution where it can successfully find a solution. The blue dotted boxes illustrate the table of which implementation reactions may be interpreted as which formal reactions, for the given node in the tree.

This completes the description of the reactionsearch algorithm. Now we turn to its correctness and its complexity. For correctness, the exhaustive depth-first search aspect of the algorithm is self-evident; the outstanding issue is whether the trivial reaction solver is guaranteed to find a solution if one exists.

Lemma 4.4. *Given a formal CRN $(\mathcal{S}, \mathcal{R})$ and implementation CRN $(\mathcal{S}', \mathcal{R}')$, let $m_0 : \mathcal{S}'' \to \mathbb{N}^{\mathcal{S}}$ be a partial interpretation on some $\mathcal{S}'' \subsetneq \mathcal{S}'$ which satisfies the atomic condition. If there exists any completion $m_1 : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ which agrees with m_0 on $\mathcal{S}''$, is a bisimulation, and is such that every implementation reaction r' involving at least one species not in $\mathcal{S}''$ has $m_1(r') = \tau$, then any minimal solution of the system of equations set up by the trivial reaction solver produces a completed interpretation m which is a bisimulation.*

Proof. It is clear that, given m_0 as described, any such m_1 will correspond to a solution of the equations set up by the trivial reaction solver on m_0. It is also clear that any m_1 produced by a solution to the trivial reaction solver equations will satisfy the atomic and delimiting conditions if and only if m_0 does (since m_0 is assumed to satisfy the atomic condition), so we assume that m_0 satisfies the delimiting condition and are concerned only with the permissive condition. We first prove that if some solution to the equations produces an m_1 that satisfies the permissive condition (thus implying that a solution exists), then there is a minimal solution to the equations that produces an interpretation m that also satisfies the permissive condition. For any formal reaction $r = R \to P$ and implementation state S' with $m(S') \geq R$ it is also true that $m_1(S') \geq m(S') \geq R$, because either m_1 is minimal or there is a minimal solution m in which each value is the same or smaller than in m_1. Since m_1 satisfies the permissive condition, there is some implementation trajectory which, under m_1, is interpreted as $S' \overset{r}{\Rightarrow}$. Since m_1 and m agree in their interpretation of every implementation *reaction*, that trajectory under m is also interpreted as $S' \overset{r}{\Rightarrow}$; since r and S' were arbitrary, m satisfies the permissive condition. Since every solution to the trivial reaction equations is $\geq$ some minimal solution, this proves that if there is any solution that produces m_1 that

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

satisfies the permissive condition, some minimal solution (in particular, the one $\leq$ it) produces m that also satisfies the permissive condition.

Having proven that at least one minimal solution produces a valid interpretation m, we show that every minimal solution does. In fact, the statement of the lemma is somewhat misleading: we show that under the above assumptions, for each formal species A that appears as a reactant in at least one formal reaction, the minimal solution to the equations for the counts of A is unique. For a formal species that never appears as a reactant, its counts in the interpretation of an implementation species cannot influence the permissive condition (given a fixed interpretation of every implementation reaction r', which is true by assumption), thus if some solution satisfies the permissive condition then every solution does. Now given m which satisfies the permissive condition, consider a complete interpretation m_2 generated by a distinct minimal solution which differs in at least one formal species that appears as a reactant. Then there is some $x \in \mathcal{S}'$ where $m(x)(A) > m_2(x(A))$ and $A \in R$ for some formal reaction $r = R \to P$ (where $m(x)(A)$ is the count of A in $m(x)$), and in particular choose x to minimize $m(x)(A)$ for the given A. Let $R_1 = R \backslash m(x)$ be the formal species in R not in $m(x)$, and let R_1' be the implementation state obtained by, for each formal species B in R_1, taking (the appropriate count of) the species x_B with $m(x_B) = m_2(x_B) = B$, which exists since the partial interpretation satisfies the atomic condition. Then for $R' = R_1' + x$, we have $m(R') \geq R$, so by the permissive condition there is a sequence of reactions which under m are interpreted as $R' \stackrel{r}{\Rightarrow}$. Let r' be the last reaction in that sequence, which means $m(r') = r$, which since all unspecified reactions must be trivial means that r' involves only species in $\mathcal{S}''$ and $m_2(r') = r$ also. By removing r', we get a sequence of reactions $R' \stackrel{\tau}{\Rightarrow} Y' + U'$, where $Y' \subset \mathcal{S}''$ and $U' \cap \mathcal{S}'' = \emptyset$. Then first of all, Y' contains all the reactants of r', and second, m and m_2 agree on every species in Y'. Since that trajectory consists of only trivial reactions under both interpretations, we must have $m(Y'+U') = m(R')$ and $m_2(Y'+U') = m_2(R')$; that is, $m(Y')+m(U') = m(R_1')+m(x)$ and $m_2(Y')+m_2(U') = m_2(R_1')+m_2(x)$. In particular, we have $m(U')(A) - m_2(U')(A) = m(x)(A) - m_2(x)(A)$. If $m(U')(A) = 0$, then this implies $m_2(U')(A) < 0$, an obvious contradiction. If $m(U')(A) > 0$, then there is some $x' \in U'$ such that $m(x')(A) > m_2(x')(A)$. But since $m(Y')(A) = R(A) > m(R')(A)$, this means $m(x')(A) \leq m(U')(A) < m(x)(A)$, which is also a contradiction since we assumed x was chosen to minimize $m(x)(A)$ of species for which $m(x)(A) > m_2(x)(A)$. So either way, given a partial interpretation which satisfies the atomic condition, if there is a complete interpretation which satisfies the three conditions in which all remaining reactions are trivial, then the trivial reaction solver will find one by searching for the first minimal solution. □

Theorem 4.4. *Given a formal CRN $(\mathcal{S}, \mathcal{R})$, implementation CRN $(\mathcal{S}', \mathcal{R}')$, and a partial interpretation which specifies $m(x)$ for some (possibly empty) set of various $x \in \mathcal{S}'$, whether a complete interpretation $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ exists that respects the given partial interpretation and is a bisimulation can be decided in polynomial space. In particular, if such an interpretation exists, then one exists that is polynomial size in that of the two CRNs and the partial interpretation.*

Proof. We prove that the reactionsearch algorithm described above outputs a correct completion of the partial interpretation if one exists and returns false if none exists; that it does so using only polynomial space; and that in particular if a correct interpretation is output then the interpretation is polynomial size in the input.

Most of the algorithm consists of, given a partial interpretation, trying out some number of ways to specify the interpretation of some species not yet specified, then calling the algorithm recursively on each of those more-specified partial interpretations. We will show that, if a correct completed interpretation exists, then at least one of those more-specified partial interpretations can be completed to that interpretation. Since the number of unspecified species and reactions decreases at each recursive call, the algorithm will eventually reach that completed interpretation. In any correct interpretation, for each formal reaction r, the atomic condition implies that there is an implementation state that interprets to the reactants of that reaction, and the permissive condition that that state must be able to eventually reach some r' with $m(r') = r$; in particular, such an r' must exist. So when the algorithm says, if no r' is known to be interpreted as r, for each possible r' enumerate all possible ways that r' can be interpreted as r, if a correct completed interpretation exists then one of those possible ways must be part of it, provided that "enumerate all possible ways" can be done in finite time and in particular in polynomial space. Similarly, by the delimiting condition every r' must have either $m(r') = r$ for some r or $m(r') = \tau$, so when the algorithm considers all those possibilities, one of them must be part of the correct completed interpretation if one exists. On the branch where the algorithm has found all r' with $m(r') \neq \tau$ in the correct interpretation, it will run the trivial reaction solver for all remaining $m(r') = \tau$. At any given time the algorithm only needs to store a single partial interpretation for each of at most $|\mathcal{S}'| + |\mathcal{R}'|$ layers of recursive calls (since each one will specify the interpretation of at least one implementation species or reaction), plus whatever information is needed to "enumerate all possible interpretations" or run the trivial reaction solver or a permissive condition test, all of which we will prove take polynomial space (with the permissive condition tests already proven). So all we have left to prove is that enumerating "all possible interpretations" of each uninterpreted species in some r' such that $m(r') = r$ for a specific r, or enumerating "all possible interpretations" for achieving the atomic condition, can be done in finite time and polynomial space, and that the trivial reaction solver works and takes polynomial space.

We first address enumerating all possible interpretations of each species in a given r' such that $m(r') = r$ for a given r. If $r' = R' \to P'$ and $r = R \to P$, then $m(r') = r$ if and only if $m(R') = R$ and $m(P') = P$. In particular, if in a partial interpretation r' can be interpreted as r, then the interpreted part of $m(R')$ must be $\leq R$ and similarly the interpreted part of $m(P')$ must be $\leq P$. By taking the difference of R minus the partial interpretation of R' and taking all assignments of each formal

species in that difference to some uninterpreted species in R', doing the same for P and P', and removing any assignments that self-contradict, we can enumerate all possible partial interpretations where $m(r') = r$. To elaborate, multiple copies of the same formal species in R or P can be assigned to different implementation species, but different copies of the same implementation species in R' and/or P' must be assigned the exact same multiset of formal species, otherwise the interpretation self-contradicts. Since all assignments of at most k copies each of i objects to j boxes can be enumerated in $\text{poly}(i, j, \log k)$ space, this process takes polynomial space. Since the interpretation of any given species involved in r' where $m(r') = r = R \to P$ must be $\leq R$ or $\leq P$, the partial interpretation throughout this entire part of the algorithm is bounded by the size of the formal CRN (except for larger interpretations provided in the initial partial interpretation), and is thus polynomial size.

Lemma 4.4 proves that if at the point when the trivial reaction solver is called a valid completion of the interpretation exists (where all remaining reactions are trivial), then the trivial reaction solver will find one and it will be a minimal solution of the given equations. It remains to show that the trivial reaction solver runs in polynomial space and produces a polynomial-size interpretation. The trivial reaction solver runs, for each formal species, the stack-based algorithm given by Contejean and Devie [40] to solve a system of linear Diophantine equations. Where q is the number of variables in the system, i.e. the number of implementation species whose interpretation is not yet specified, Contejean and Devie prove that their algorithm stores at most q states at one time on the stack, each of which is a tuple of q integers. Pottier has proven a bound on the size of those integers, namely, that their sum is at most, in that notation, $(1 + ||A||_{1,\infty})^r$ [41]. There r is bounded above by the number of unknown implementation reactions and $||A||_{1,\infty}$ is the maximum of any individual equation (i.e., unknown implementation reaction) of the sum of coefficients in that equation (coefficients of unknown implementation species in the reaction, or of formal species in the known interpretations involved). The "size" of the given interpretation is bounded by q times the logarithm of the bound on an individual count, specifically, $|m| \leq qr(1 + \log ||A||_{1,\infty})$, where $\log ||A||_{1,\infty}$ itself is the "size" of some combination of the implementation CRN and the current partial interpretation. The implementation CRN is of course part of the input to the algorithm, and we have proven that the partial interpretation up to this point is polynomial in the size of the input; thus the entire algorithm runs in space polynomial in its input, and if a correct interpretation exists, then one exists which is polynomial size. □

In the general case, finding an interpretation turns out to be just as hard as checking an interpretation; in fact, the same space-bounded Turing machine reduction from Theorem 4.3 applies.

Theorem 4.5. *Whether a bisimulation interpretation exists from a given implementation CRN to a given formal CRN is PSPACE-complete.*

Proof. Theorem 4.4, with Theorem 4.1 for checking the permissive condition, prove that a bisimulation can be found, or shown that none exists, in polynomial space. To prove completeness, we use the same formal and implementation CRN used in Theorem 4.3 and shown in Fig. 12. Consider an arbitrary Turing machine with m states and tape alphabet $\{0, 1\}$, with start state q^0 and halt state q^{m-1} which on any input, halts without using more space than the length of the input, with the tape reading 10^{n-1} if it accepts and 0^n if it rejects; also consider an input x with length n. Given that, the formal CRN has $n + 2$ species and 1 reaction, $Q + A_1 + \cdots + A_n \to H$. The implementation CRN has species 0_i and 1_i for each tape spot i, q_i^j for each tape spot i and each Turing machine state j, additional q_i^m for a "reset" state m and each tape spot, and a halt species h. As described in Theorem 4.3, the implementation CRN can simulate the Turing machine with reactions $q_i^j + \sigma_i \to q_{i\pm1}^{j'} + \sigma_i'$; can "reset" the computation to the start state reading string x with reactions $q_i^j \to q_n^m$ (for all q_i^j except q_i^{m-1} for $i > 1$), $q_i^m + \sigma_i \to q_{i-1}^m + x_i$, and $q_1^m + \sigma_1 \to q_0^1 + x_1$; and can check whether the computation has accepted with reactions $q_1^{m-1} + 1_1 \rightleftharpoons q_2^{m-1}$, $q_{i-1}^{m-1} + 0_{i-1} \rightleftharpoons q_i^{m-1}$ for $i > 2$, and $q_n^{m-1} + 0_n \to h$. We showed in Theorem 4.3 that the interpretation $m(0_i) = m(1_i) = A_i$, $m(h) = H$, $m(q_i^j) = Q$ for $j \neq m - 1$ and $m(q_i^{m-1}) = Q + \sum_{k<i} A_k$ is valid if and only if the given Turing machine accepts the string x, thus proving that checking an interpretation is PSPACE-complete. To prove that finding an interpretation is PSPACE-complete, we show that aside from permutations of the formal species in that interpretation (which are also correct if and only if the Turing machine accepts x), no other interpretation can be correct; therefore, a correct interpretation exists if and only if the Turing machine accepts x.

To prove that only one correct interpretation (up to permutations) is possible, we use the three conditions to eliminate possibilities until only that one remains. First, by applying the delimiting condition to the reactions $q_i^j \to q_n^m$, either all q_i^j (except q_i^{m-1} for $i > 1$) must have the same interpretation, or some subset of them (including q_n^m) must interpret to H and the rest to $Q + A_1 + \cdots + A_n$. We can quickly eliminate the cases $m(q_n^m) = \emptyset$ and $m(q_n^m) = H$, and prove that $m(0_i) = m(1_i)$ for every i.

If $m(q_n^m) = \emptyset$, then by the atomic condition there must be some state which interprets to exactly $Q + A_1 + \cdots + A_n$ by, for each of those formal species, selecting one implementation species that interprets to exactly one of that formal species and nothing else. For each q_i^j that appears, apply the appropriate sequence of reactions from $q_i^{m-1} \to q_{i-1}^{m-1} + 0_{i-1}$, $q_2^{m-1} \to q_1^{m-1} + 1_1$, and $q_i^j \to q_n^m$ until the only q_i^j that appears is some number of copies of q_n^m. All these reactions must be trivial (or the delimiting condition is violated): each of the q_i^{m-1}-involving reactions are reversible while the formal reaction

is not, while if $m(q_n^m) = \emptyset$ then any $q_i^j \to q_n^m$ reaction has product $\emptyset$ while no formal reaction does; thus the resulting implementation state has the same interpretation as the original, namely $Q + A_1 + \cdots + A_n$. Since $m(q_n^m) = \emptyset$, by removing all copies of it we get another implementation state with interpretation $Q + A_1 + \cdots + A_n$ but no copies of q_i^j for any i, j. Since every implementation reaction has some q_i^j as a reactant, no reactions can fire in this state and the permissive condition is violated.

If $m(q_n^m) = H$ then $m(q_i^m) = m(q_1^0) = H$ because the "reset" reactions $q_i^m + x_i \to q_{i-1}^m + x_i$ and $q_1^m + x_1 \to q_1^0 + x_1$ (using the notation x_i for 0_i if the ith symbol of the string x is 0 and 1_i if it is 1, and x_{-i} for 1_i if the ith symbol of x is 0 and 0_i if it is 1; these are the reactions used in the event that the tape already has the correct symbol) must be trivial (since no formal reaction has an H as a reactant), and the only difference between the two sides is the two q's. Since if $m(q_n^m) \neq H$ we know $m(q_i^m) = m(q_1^0) = m(q_n^m)$ anyway, the reset reactions $q_i^m + x_{-i} \to q_{i-1}^m + x_i$ and $q_1^m + x_{-1} \to q_1^0 + x_1$ prove that $m(0_i) = m(1_i)$ for all i: the reaction must be trivial since $m(q_n^m) \neq \emptyset$ and no formal reaction is catalytic. By the atomic condition, there are $n+1$ species $Q, A_1, \ldots, A_n$ which must each have some implementation species that interprets as one copy of that species, and since $m(0_i) = m(1_i)$ the 0_i's and 1_i's can account for at most n of them. Call the remaining species X, and note that by assumption either $m(q_1^{m-1}) = H$ or $m(q_1^{m-1}) = Q + A_1 + \cdots + A_n$. If $m(h) = X$, then the sequence of reactions $q_1^{m-1} + 1_i \to q_2^{m-1}$, $q_{i-1}^{m-1} + 0_{i-1} \to q_i^{m-1}$ for $3 \leq i \leq n$ in order, then $q_n^{m-1} + 0_n \to h$ is a sequence which takes a state whose interpretation contains either an H or *all* of $Q, A_1, \ldots, A_n$ to a state which has no H and only one of $Q, A_1, \ldots, A_n$; this is impossible in the formal CRN, thus at least one implementation reaction on that pathway must be nontrivial and not a formal reaction. The only remaining implementation species are the q_i^j's, and we have already shown that if $m(q_n^m) = H$ then $m(q_i^j)$ for every $j \neq m$ is either H or $Q + A_1 + \cdots + A_n$, and cannot be just X. That leaves only some q_i^{m-1} for $i \geq 2$, but since the reversible reactions must be trivial, $m(q_i^{m-1}) = m(q_1^{m-1}) + \sum_{k<i} m(0_k)$, which must contain either an H or all of $Q + A_1 + \cdots + A_n$, and cannot be just X. Thus it is impossible to have $m(q_n^m) = H$, and we must have $m(q_i^j) = m(q_n^m)$ whenever $j \neq m-1$.

So far we know that $m(0_i) = m(1_i)$ for all i, $m(q_n^m)$ is neither $\emptyset$ nor H, $m(q_i^j) = m(q_n^m)$ for $j \neq m-1$, and since the reversible reactions must be trivial $m(q_i^{m-1}) = m(q_n^m) + \sum_{k<i} m(0_i)$. In order to satisfy the atomic condition for $n+2$ formal species, we need $n+2$ implementation species with distinct interpretations. Regardless of the interpretations of any other species, $m(q_i^{m-1}) \geq m(q_n^m)$ so no q_i^{m-1} can be interpret as a single formal species other than the one (if any) that q_n^m is interpreted as, so all the q_i^j's can satisfy the atomic condition for at most one formal species. Each pair of 0_i and 1_i must have the same interpretation, so all the 0_i's and 1_i's can satisfy at most n species, and h can satisfy an additional one, but no other implementation species remain. So since we have $n+2$ "categories" of implementation species that can possibly be interpreted as a single formal species and $n+2$ formal species, each such group must be interpreted as a distinct formal species. Given that, the reaction $q_n^{m-1} + 0_n \to h$ cannot be trivial, so it must be interpreted as $Q + A_1 + \cdots + A_n \to H$; in particular, we must have $m(h) = H$. The remaining constraints say that each 0_i and q_n^m is interpreted as a distinct one of Q or some A_i, that $m(1_i) = m(0_i)$, that $m(q_i^j) = m(q_n^m)$ for $j \neq m-1$, and that $m(q_i^{m-1}) = m(q_n^m) + \sum_{k<i} m(0_i)$; this is exactly the interpretation given in Theorem 4.3, up to a permutation of the formal species Q and the A_i's. Any such interpretation will satisfy the atomic and delimiting conditions, and will satisfy the permissive condition if and only if the given Turing machine accepts the string x, thus finding a correct interpretation is as hard as deciding whether a linear bounded Turing machine accepts a given string, which is PSPACE-complete. □

When the number of formal reactants is bounded by a constant, we showed that whether an interpretation is valid can be checked in polynomial time. Then finding an interpretation is a natural NP problem; we show that it is in fact NP-complete, with a reduction from 3-SAT. An example of this reduction is shown in Fig. 14.

Theorem 4.6. *When the number of reactants in a formal reaction k is bounded by a constant $k \geq 1$, whether a bisimulation interpretation exists is NP-complete.*

Proof. If a valid interpretation exists, Theorem 4.4 guarantees that we can find a valid polynomial-size interpretation which can be checked in polynomial time by Theorem 4.2.

To prove NP-completeness, given an arbitrary 3-SAT formula we construct a formal and implementation CRN such that a valid interpretation exists if and only if the formula is satisfiable. Our formal CRN has two species C and T and three reactions $C \to T$, $C \to 2T$, and $C \to 3T$. Our implementation CRN has two species s_C and s_T plus for each variable x_i in the 3-SAT formula two species x_i^t and x_i^f. We encode each clause of our 3-SAT formula e.g. $(x_1 \vee \neg x_2 \vee x_3)$ as an implementation reaction e.g. $s_C \to x_1^t + x_2^f + x_3^t$. Formally, we say that a clause is $(l_1 \vee l_2 \vee l_3)$ where each l_j is some x_i or $\neg x_i$, and it is encoded in the implementation reaction $s_C \to v_1 + v_2 + v_3$, where v_j is x_i^t if l_j is x_i, or x_i^f if l_j is $\neg x_i$. We also add the implementation reactions $s_C \to s_T$, $s_C \to 2s_T$, and $s_C \to 3s_T$, and for each x_i the reaction $s_T \rightleftharpoons x_i^t + x_i^f$ which we will show restrict interpretations that satisfy the three conditions to correspond to satisfying assignments of the 3-SAT formula.

We again observe that none of the formal reactions are reversible, so in order to satisfy the delimiting condition the reactions $s_T \rightleftharpoons x_i^t + x_i^f$ must be trivial. Note that all other implementation reactions have exactly s_C as their reactants. The

3-SAT problem	Formal CRN	Implementation CRN
	$C \to T$	$s_C \to s_T$
	$C \to 2T$	$s_C \to 2s_T$
	$C \to 3T$	$s_C \to 3s_T$
$(x_1 \vee \neg x_2 \vee x_3)$	$(C \to 2T)$	$s_C \to x_1^t + x_2^f + x_3^t$
$\wedge(\neg x_1 \vee x_4 \vee \neg x_2)$	$(C \to 2T)$	$s_C \to x_1^f + x_4^t + x_2^f$
$\wedge(x_2 \vee \neg x_3 \vee \neg x_4)$	$(C \to T)$	$s_C \to x_2^t + x_3^f + x_4^f$
	$(T \rightleftharpoons T)$	$s_T \rightleftharpoons x_1^t + x_1^f$
	$(T \rightleftharpoons T)$	$s_T \rightleftharpoons x_2^t + x_2^f$
	$(T \rightleftharpoons T)$	$s_T \rightleftharpoons x_3^t + x_3^f$
	$(T \rightleftharpoons T)$	$s_T \rightleftharpoons x_4^t + x_4^f$

satisfying assignment	valid interpretation
$x_1 = \text{true}$	$m(x_1^t) = T,\ m(x_1^f) = \emptyset$
$x_2 = \text{false}$	$m(x_2^t) = \emptyset,\ m(x_2^f) = T$
$x_3 = \text{false}$	$m(x_3^t) = \emptyset,\ m(x_3^f) = T$
$x_4 = \text{true}$	$m(x_4^t) = T,\ m(x_4^f) = \emptyset$
	$m(s_C) = C,\ m(s_T) = T$

Fig. 14. Example formal and implementation CRN corresponding to an instance of the 3-SAT problem. Top left: example 3-SAT instance. Top middle: the formal CRN. Eventual interpretations of the corresponding implementation reactions are given in parentheses. Top right: the implementation CRN corresponding to this 3-SAT instance. Each 3-SAT clause has a corresponding implementation reaction, with auxiliary implementation reactions added. Bottom left: a satisfying assignment for the 3-SAT formula. Bottom right: a valid CRN bisimulation interpretation for the two CRNs, which the reactionsearch algorithm would find. Note the correspondence between satisfying assignment and interpretation: if x_i is true then $m(x_i^t) = T$ and $m(x_i^f) = \emptyset$, otherwise $m(x_i^t) = \emptyset$ and $m(x_i^f) = T$. Such an interpretation is a CRN bisimulation if and only if the corresponding assignment satisfies the 3-SAT formula; if no satisfying assignment exists, then no valid interpretation exists.

combination of atomic and permissive conditions implies that each formal reaction must have at least one implementation reaction that is interpreted as it; since all three formal reactions have reactants C and all remaining implementation reactions have reactants s_C, any valid interpretation must have $m(s_C) = C$. Now the reactions $s_C \to s_T$ and $s_C \to 2s_T$ cannot both be trivial, since that would imply $m(s_T) = 2m(s_T)$ and thus $m(s_T) = \emptyset$, which from the reactions $s_T \rightleftharpoons x_i^t + x_i^f$ implies $m(x_i^t) = m(x_i^f) = \emptyset$, leaving not enough species to satisfy the atomic condition. Therefore at least one of those two reactions must be formal; the only way to satisfy that is $m(s_T) = T$, making those two reactions and $s_C \to 3s_T$ interpreted as the three formal reactions respectively, and also satisfying the atomic condition.

Given that $m(s_C) = C$ and $m(s_T) = T$, since the reversible reactions are all trivial we have for each i, $m(x_i^t) + m(x_i^f) = T$. In other words, exactly one of $m(x_i^t)$ or $m(x_i^f)$ is T, and the other is $\emptyset$. Such an interpretation satisfies the atomic condition with $m(s_C) = C$ and $m(s_T) = T$, and satisfies the permissive condition since any state whose interpretation has a C has an s_C, and can do the reactions $s_C \to s_T$, $s_C \to 2s_T$, or $s_C \to 3s_T$ for whichever formal reaction is desired; further, those three reactions and the reversible $s_T \rightleftharpoons x_i^t + x_i^f$ satisfy the delimiting conditions, leaving only the reactions for each clause of the 3-SAT formula. These interpretations have an obvious one-to-one correspondence with assignments to the variables in the 3-SAT formula: x_i is assigned to true if $m(x_i^t) = T$, or to false if $m(x_i^f) = T$. Each reaction $s_C \to v_1 + v_2 + v_3$, corresponding to a clause $(l_1 \vee l_2 \vee l_3)$, will be interpreted as either $C \to \emptyset$, $C \to T$, $C \to 2T$, or $C \to 3T$, depending on how many of l_1, l_2, and l_3 are true in the corresponding assignment. Specifically, the interpretation will satisfy the delimiting condition (none of the clause reactions are interpreted as $C \to \emptyset$) if and only if the assignment satisfies the formula (none of the clauses have no true variables). Thus a valid interpretation exists if and only if the formula has a satisfying assignment, which completes the proof of NP-completeness. □

4.3. Using the modularity condition

Because finding and checking interpretations are in some cases computationally intractable, any way of reducing the size of the problem would be helpful. Often, a larger implementation CRN can be broken up into a number of smaller modules. The modularity condition of Definition 3.3 shows that each module can be checked individually and combined into a correct implementation, as described in Theorem 3.2 and Corollary 3.3. We show that the reactionsearch algorithm from Section 4.2

can be modified to iterate through a number of correct interpretations such that if any correct interpretation is modular with respect to the given sets of common formal and implementation species, then one of the enumerated interpretations is. We also show that whether an interpretation is modular with respect to given sets of common formal and implementation species can be checked in polynomial time, and present an algorithm to do so. We begin by proving Lemma 4.5, which provides the mathematical foundation for the modified reactionsearch algorithm. Then, Lemma 4.6 presents an algorithm for testing modularity and establishes its running time. Finally, Theorem 4.7 presents the modified reactionsearch algorithm for finding a modular interpretation and establishes its running time. Note that the user will be responsible for identifying the modules and the common species; the modified reactionsearch algorithm will be responsible for finding a valid modular interpretation with respect to those modules, if one exists. Thus, below, the formal CRN and implementation CRN that we discuss will be only those species and reactions relevant to verification of a single module at a time.

Recall that the reactionsearch algorithm for finding an interpretation was based on Lemma 4.4 which proved that if a valid completion of an interpretation exists when the trivial reaction solver is called, the trivial reaction solver will find a valid completed interpretation. Our first step is to show that if at the same point a valid *modular* completion exists, then the trivial reaction solver will find one.

Lemma 4.5. *Given a formal CRN $(\mathcal{S}, \mathcal{R})$, implementation CRN $(\mathcal{S}', \mathcal{R}')$, and sets of common species $\mathcal{S}_0$ and $\mathcal{S}'_0$, let $m_0 : \mathcal{S}'' \to \mathbb{N}^{\mathcal{S}}$ be a partial interpretation defined on some set of implementation species $\mathcal{S}''$ with $\mathcal{S}'_0 \subset \mathcal{S}'' \subsetneq \mathcal{S}'$, where m_0 satisfies the atomic condition. If there exists any completion $m_1 : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ that agrees with m_0 on S'', is a modular bisimulation with respect to $\mathcal{S}_0$ and $\mathcal{S}'_0$, and such that every reaction r' involving at least one species not in S'' has $m_1(r') = \tau$, then any minimal solution of the system of equations set up by the trivial reaction solver produces a completed interpretation m which is a modular bisimulation with respect to $\mathcal{S}_0$ and $\mathcal{S}'_0$. If every formal species in $\mathcal{S}_0$ appears as a reactant in $\mathcal{R}$, then the previous statement holds even if $\mathcal{S}'_0 \not\subset \mathcal{S}''$.*

As a remark, we give two conditions of which only one is required for the lemma to hold: $\mathcal{S}'_0 \subset \mathcal{S}''$ (the interpretation of the common implementation species is already known) *or* every formal species in $\mathcal{S}_0$ appears in a reaction. In the typical use case of this algorithm, namely a systematic DNA strand displacement implementation to which we want to apply Corollary 3.3, the first condition holds but the second does not. In such a system, $\mathcal{S}'_0$ is typically the set of all signal species $\{x_A, x_B, \dots\}$, for which we "know" that $m(x_A) = A$ etc., at least in the sense that any interpretation where that is not true is not interesting. However, in such a system $\mathcal{S}_0 = \mathcal{S}$ is the set of all formal species, in that each module officially contains all formal species of the larger CRN even if only some of them appear in a reaction in any given module.

Proof. First, we show that if some solution to the trivial reaction solver equations produces a modular bisimulation, then some minimal solution does. Let m_1 be the modular bisimulation produced by the solution that exists by assumption, and let m be the interpretation produced by an arbitrary minimal solution $\leq$ that solution; so for all x, $m(x) \leq m_1(x)$. Lemma 4.4 shows that m is a bisimulation, so we only need to prove it is modular. Given any x, there is a sequence of trivial reactions (since any two solutions agree on the interpretation of any reaction, the sequence is trivial under both m_1 and m) by which $x \stackrel{\tau}{\Rightarrow} Y + Z$, where $Y \subset \mathcal{S}'_0$ and $m_1(Z) \cap \mathcal{S}_0 = \emptyset$. Since $m \leq m_1$, in particular $m(Z) \leq m_1(Z)$ and $m(Z) \cap \mathcal{S}_0 = \emptyset$, so the same sequence of trivial reactions satisfies the modularity condition for m. Since this applies for all x, m is also modular.

Now that there exists at least one minimal solution which produces a modular bisimulation, let m be that modular bisimulation and m_2 a bisimulation produced by another minimal solution. Given that the interpretation of every reaction is the same, we know we can treat the solution for each formal species separately. For species $B \notin \mathcal{S}_0$, if $x \stackrel{\tau}{\Rightarrow} Y + Z$ for $Y \subset \mathcal{S}'_0$ and $m(Z) \cap \mathcal{S}_0 = \emptyset$ then m_2 disagreeing with m on the count of B in any species will not affect whether $m_2(Z) \cap \mathcal{S}_0 = \emptyset$; if m and m_2 only disagree on species not in $\mathcal{S}_0$ then m_2 is modular. (Since Lemma 4.4 proves that if any minimal solution produces a bisimulation then the minimal solution is unique on all formal species that appear as a reactant, if every species in $\mathcal{S}_0$ appears in a reaction then the above completes the proof without requiring $\mathcal{S}'_0 \subset \mathcal{S}''$.) If m and m_2 disagree on any $m(x)(A)$ for $A \in \mathcal{S}_0$, then take any such x look at the reactions by which $x \stackrel{\tau}{\Rightarrow} Y + Z$ with $Y \subset \mathcal{S}'_0$ and $m(Z) \cap \mathcal{S}_0 = \emptyset$. Now using the assumption $\mathcal{S}'_0 \subset \mathcal{S}''$ so m and m_2 agree on any species in $\mathcal{S}'_0$, in particular $m(x)(A) = m(Y)(A) = m_2(Y)(A)$, which since $m(Z)(A) = 0$ implies $m_2(Z)(A) = m_2(x)(A) - m(x)(A) > 0$. Since each formal species can be taken independently, if $m_2(x)(A) > m(x)(A)$ whenever they disagree then an interpretation m_3 defined as $m_3(x)(A) = m(x)(A)$ and $m_3(x)(B) = m_2(x)(B)$ for $B \neq A$ is also a solution, but $m_3 \leq m_2$ so m_2 is not minimal, a contradiction. This proves that the minimal solution to the trivial reaction solver is unique and correct for any formal species A that appears as a reactant in a formal reaction (Lemma 4.4) or is in $\mathcal{S}_0$ (assuming $\mathcal{S}'_0 \subset \mathcal{S}''$), and any other formal species can't affect whether the interpretation is a modular bisimulation; so if a modular bisimulation exists, then given the conditions, any minimal solution to the trivial reaction equations produces one. □

Given a formal and implementation CRN $(\mathcal{S}, \mathcal{R})$ and $(\mathcal{S}', \mathcal{R}')$, an interpretation m which we already know is a bisimulation, and sets of common species $\mathcal{S}_0$ and $\mathcal{S}'_0$, we can check whether m is modular with respect to $\mathcal{S}_0$ and $\mathcal{S}'_0$ using an algorithm similar to the graphsearch algorithm for checking the permissive condition. First, construct a table which for each implementation species $x \in \mathcal{S}'$ stores either that x is "finished", or if it is not finished, stores a list of all $x' \in \mathcal{S}'$ such that $m(x') = x$ and $x \stackrel{\tau}{\Rightarrow} x' + Z$ is known (x "can reach" x') and a list of all $z \in \mathcal{S}'$ such that $x \stackrel{\tau}{\Rightarrow} x + z$ is known (x "can produce" z;

this implies $m(z) = \emptyset$). Here "x is finished" means either it is known that $x \overset{\tau}{\Rightarrow} Y + Z$ for $Y \subset \mathcal{S}_0'$ and $m(Z) \cap \mathcal{S}_0 = \emptyset$, or it is known that $x \overset{\tau}{\Rightarrow} x_1' + X'$ for $\emptyset < m(x_1') < m(x)$. We will show later that if all x meet one of those two conditions, then m is modular with respect to $\mathcal{S}_0$ and $\mathcal{S}_0'$. Initialize the table to say that x is finished if $x \in \mathcal{S}_0'$ or $m(x) \cap \mathcal{S}_0 = \emptyset$, otherwise x is known to reach itself and not known to produce anything. Then in cycles, for each x not yet finished where Z_x is the set of all z such that $x \overset{\tau}{\Rightarrow} x + z$ is known, for each trivial reaction that can happen in a state with one x and arbitrarily many copies of Z_x (written $x + \infty Z_x \xrightarrow{\tau} \dots$), update the table according to the following rules:

(i) If $x + \infty Z_x \xrightarrow{\tau} Y$ where every species in Y is finished, then x is finished.
(ii) If $x + \infty Z_x \xrightarrow{\tau} x_1' + X'$ where $\emptyset < m(x_1') < m(x)$, then x is finished.
(iii) If $x + \infty Z_x \xrightarrow{\tau} x' + Z$ where $m(x') = m(x)$ (implying $m(Z) = \emptyset$), then x can reach x' and x can reach any species that x' can reach.
(iv) If $x + \infty Z_x \xrightarrow{\tau} x' + Z$ where $m(x') = m(x)$ and x' can reach x, then x can produce any species $z \in (Z \cup Z_{x'})$.

Continue until one cycle (checking every trivial reaction for every unfinished x) passes with no changes to the table. At that time, if every x is finished then m is modular, otherwise m is not modular (with respect to $\mathcal{S}_0$ and $\mathcal{S}_0'$).

Lemma 4.6. *Given a formal and implementation CRN $(\mathcal{S}, \mathcal{R})$ and $(\mathcal{S}', \mathcal{R}')$, a bisimulation $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$, and common species sets $\mathcal{S}_0 \subset \mathcal{S}$ and $\mathcal{S}_0' \subset \mathcal{S}'$, whether m is modular with respect to $\mathcal{S}_0$ and $\mathcal{S}_0'$ can be checked in polynomial time.*

Proof. We prove that the above algorithm is correct and runs in polynomial time. For correctness, first, the table is initialized with true facts and updated with true deductions: for initialization, if $x \in \mathcal{S}_0'$ then $x \overset{\tau}{\Rightarrow} x + \emptyset$, and if $m(x) \cap \mathcal{S}_0 = \emptyset$ then $x \overset{\tau}{\Rightarrow} \emptyset + x$, as sequences of 0 reactions fulfill $x \overset{\tau}{\Rightarrow} Y + Z$ for $Y \subset \mathcal{S}_0'$ and $m(Z) \cap \mathcal{S}_0 = \emptyset$. The deductions in general follow from that, if $x \overset{\tau}{\Rightarrow} x + Z_x$ and $x + \infty Z_x \xrightarrow{\tau} S'$, then $x \overset{\tau}{\Rightarrow} S'$. The only non-obvious one is that if $x \overset{\tau}{\Rightarrow} Y'$ where every species in Y' is finished, then x is finished; this follows from induction on the order in which the algorithm marks species as finished: if every species x' marked as finished before x satisfies one of the two $x' \overset{\tau}{\Rightarrow} \dots$ conditions in the definition of finished, and $x \overset{\tau}{\Rightarrow} Y'$ made up of only those species, then x satisfies one of the two conditions. Then we need to prove that if every $x \in \mathcal{S}'$ is finished then every $x \overset{\tau}{\Rightarrow} Y + Z$ as desired: the proof is by induction on $|m(x)|$, proving that if $x' \overset{\tau}{\Rightarrow} Y' + Z'$ for every x' with $m(x') < m(x)$ and x is finished then $x \overset{\tau}{\Rightarrow} Y + Z$. Recall that "$x$ is finished" is defined as, either $x \overset{\tau}{\Rightarrow} Y + Z$ for $Y \subset \mathcal{S}_0'$ and $m(Z) \cap \mathcal{S}_0 = \emptyset$, or $x \overset{\tau}{\Rightarrow} x_1' + X'$ for $\emptyset < m(x_1') < m(x)$. If $x \overset{\tau}{\Rightarrow} Y + Z$ then we are done. If $x \overset{\tau}{\Rightarrow} x_1' + X'$ for $\emptyset < m(x_1') < m(x)$, then every species $x' \in X'$ has $m(x') \leq m(x) - m(x_1') < m(x)$, so by the induction hypothesis every $x' \in X' \cup \{x_1'\}$ is finished and therefore has $x' \overset{\tau}{\Rightarrow} Y_{x'} + Z_{x'}$ as desired, so by combining all those $x \overset{\tau}{\Rightarrow} \sum_{x'} Y_{x'} + \sum_{x'} Z_{x'}$, satisfying the modularity condition. This proves that if the algorithm says m is modular, then it is.

To complete the proof that the algorithm is correct, we show that if m is in fact modular, the algorithm will not say it is not. The algorithm terminates when a cycle passes with no change to the table, so we show that if m is modular but at the beginning of a cycle at least one species is not yet finished, then there is some fact not yet in the table that will be learned this cycle. Consider a modified implementation CRN where the reaction $x \xrightarrow{\tau} x + Z_x$ for the current Z_x known to be producible at x is added for each x; if m is modular in the original CRN then it is modular in the new CRN, since reactions were only added. Then consider, for each $x \in \mathcal{S}'$, the first reaction on the path with the fewest non-x $x \xrightarrow{\tau} x + Z_x$ reactions by which $x \overset{\tau}{\Rightarrow} Y + Z$ to satisfy the modularity condition; the algorithm will consider these reactions (among others) as possible and update based on them this cycle. Now consider the path obtained by starting from an x not yet finished, at any given state $\{|x'|\}$ (ignoring null species), taking that first reaction, until either a $Y + Z$ satisfying the modularity condition is reached, or some $x_1' + X'$ with $\emptyset < m(x_1') < m(x)$ is reached, or a non-null species is repeated; for this purpose, the reactions $x' \xrightarrow{\tau} x' + Z_{x'}$ do not count as repeating a species. (Given finitely many x' with $m(x') = m(x)$, one of those three must eventually happen.) If the path ends in $Y + Z$ or $x_1' + X'$, then the last x' with $m(x') = m(x)$ along the path which is not yet finished (which may be x) will be marked as finished this cycle. If the path ends by repeating a species, say $x \overset{\tau}{\Rightarrow} x_0' + Z_1 \overset{\tau}{\Rightarrow} x_0' + Z_1 + Z_0$ with $m(x_0') = x$, then if any species x_1' in the loop $x_0' \overset{\tau}{\Rightarrow} Z_1 + Z_0$ is not known to reach some other species x_2' in the loop, for each x_2' the last such species will become known to reach x_2' this cycle. If every species in the loop is known to reach every other species in the loop, and x_0' is known to produce every species in Z_0, then replacing the $x_0' \overset{\tau}{\Rightarrow} x_0' + Z_0$ loop with the (in our measure of path length, 0-length) reaction $x_0' \xrightarrow{\tau} x_0' + Z_{x_0'}$ would create a shorter path by which $x \overset{\tau}{\Rightarrow} Y + Z$; so there is some $z \in Z_0$ not known to be produced from x_0'. Somewhere in that loop is a reaction $x_1' \xrightarrow{\tau} x_2' + z + Z_2$, and since every species in the loop is known to reach every other species in the loop (including itself), the last species in the loop before x_1' which is not yet known to produce z (which may be x_0') will be known to produce z this cycle. This covers all cases, and completes this part of the proof: if m is modular but the algorithm does not yet known that every species is finished, it will learn at least one new fact each cycle, thus never saying no.

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

To complete the proof, we show that the algorithm always terminates in polynomial time. Each cycle consists of for each implementation species, for each trivial reaction, checking whether that reaction is possible from that species plus null species, checking properties in the table for each of the produced species, and updating the table, a polynomial number of polynomial-time operations. Since at least one fact must be learned each cycle, the number of cycles is bounded by the number of facts: $n(n+z+1)$, where n is the number of implementation species and z the number of null species, so the algorithm is guaranteed to terminate in polynomial time. Since the algorithm is guaranteed to terminate, that the algorithm never returns no when m is modular implies that it returns yes, which also completes the proof of correctness. □

To find modular bisimulation interpretations, we modify the reactionsearch algorithm such that after checking the permissive condition it also uses the above algorithm to check the modularity condition. For this to be correct we would need to prove that if a modular bisimulation exists then the algorithm will find it; thankfully this is true.

Theorem 4.7. *Given a formal and implementation CRN $(\mathcal{S},\mathcal{R})$ and $(\mathcal{S}',\mathcal{R}')$, sets of common species $\mathcal{S}_0\subset\mathcal{S}$ and $\mathcal{S}'_0\subset\mathcal{S}'$, and a partial interpretation which specifies $m(x)$ for some set $\mathcal{S}''\subset\mathcal{S}'$ of various $x\in\mathcal{S}''$, provided that either $\mathcal{S}'_0\subset\mathcal{S}''$ or every formal species in $\mathcal{S}_0$ appears as a reactant in $\mathcal{R}$, whether a complete interpretation $m:\mathcal{S}'\to\mathbb{N}^{\mathcal{S}}$ exists that respects the given interpretation and is a modular bisimulation with respect to $\mathcal{S}_0$ and $\mathcal{S}'_0$ can be decided in polynomial space. In particular, if such an interpretation exists, then the modified reactionsearch algorithm will find one that is polynomial size in that of the two CRNs and the partial interpretation.*

Proof. That the modified reactionsearch algorithm outputs only polynomial-size bisimulations and runs in polynomial space is proven in Theorem 4.4, so when combined with the modularity checker it will output only polynomial-size modular bisimulations. If a complete modular bisimulation exists, then whatever partial interpretation of it specifies the interpretation of all nontrivial reactions will be considered by the modified reactionsearch algorithm, at which point it will call the trivial reaction solver; so far, the proof is the same as Theorem 4.4. Given that partial completion, by Lemma 4.5, if a completion of it exists (which it does) then one exists which is a minimal solution of the trivial reaction solver equations. That such an interpretation must be polynomial-size is again proven in Theorem 4.4, so it will be found and verified by the permissive and modularity checkers, and returned. □

Given a large formal CRN and implementation CRN along with a user-provided breakdown into modules with defined common species and a partial interpretation on the common implementation species, the modified reactionsearch algorithm may be applied sequentially (or in parallel) to each module; if all modules admit a valid modular interpretation, then a valid interpretation for the full system exists—specifically, the union of all the modules' interpretations, which will be consistent since they share the given interpretation on the common species. Furthermore, if the algorithm fails to find an interpretation for some module, then no valid modular interpretation exists (although it remains possible that a non-modular valid interpretation exists). Note that the choice of modules must be consistent with the requirements of Theorem 3.2, for example, every module contains the same set of common species and their intersection contains no other species.

We have shown that finding a modular bisimulation is not significantly harder than finding a bisimulation at all, and in fact finding a modular bisimulation for a large CRN broken into many modules is much easier than trying to find a bisimulation for the whole CRN with no information about its modularity. On a trivial level, any interpretation is modular with respect to common implementation species $\mathcal{S}'_0=\mathcal{S}'$ regardless of the common formal species *or* common formal species $\mathcal{S}_0=\emptyset$ regardless of the common implementation species. Thus, Theorems 4.3, 4.5, and 4.6 apply, and checking a modular bisimulation is PSPACE-complete in general (but polynomial time in n^k if the largest number of reactants in a formal reaction is k), and finding one is PSPACE-complete in general and NP-complete when the number of reactants in a formal reaction is bounded by a constant. Whether this stays true for more "meaningful" cases of modularity is a more interesting question. For example, the property $\mathcal{S}_0=\mathcal{S}$, $|\mathcal{S}'_0|=|\mathcal{S}_0|$ and satisfies the atomic condition—every formal species is common, and the common implementation species consist of exactly one species x_A with $m(x_A)=A$ for each formal species A—describes the typical non-history-domain systematic DSD implementation of CRNs, and neither of the CRNs in Theorems 4.3, 4.5, or 4.6 are modular with respect to any sets with that property. Whether there is another hardness proof for that sort of set of common species, or whether checking or finding a modular interpretation is in fact easier in (the worst case of) that subcase, is currently an open question.

5. Additional features of CRN bisimulation

5.1. Bisimulation in transition systems

We call this theory "CRN bisimulation" because it is a special case of the theory of weak bisimulation in concurrent systems, adapted to CRNs. In [19], this theory is defined in terms of a *labeled transition system*

$$(\Sigma,\mathcal{T},\xrightarrow{t}:t\in\mathcal{T})$$

where Σ is a set of *states*, $\mathcal{T}$ a set of *labels*, and for each $t \in \mathcal{T}$ there is a relation $\xrightarrow{t} \subset \Sigma \times \Sigma$, specifying which states can transition to which other states by an *action* of type t. For example, a CRN $(\mathcal{S}, \mathcal{R})$ can be expressed as a labeled transition system; there are multiple ways to do this, but to give one particularly natural way, let $\Sigma = \mathbb{N}^{\mathcal{S}}$ be the set of all states in the usual sense of the CRN, $\mathcal{T} = \mathcal{R}$ so that the labels for transitions are the reactions, and for $r = R \to P \in \mathcal{R}$ the transition relation is $\xrightarrow{r} = \{(S, S - R + P) \mid S \in \mathbb{N}^{\mathcal{S}}, S \geq R\}$. This construction matches the semantics of CRNs as defined in Section 2, and is the basis of the connection between our theory of CRN bisimulation and weak bisimulation as defined in [19].

Aside from labeled transition systems, however, the paradigm used by Milner in [19] diverges from the paradigm we use when discussing CRNs. Milner discusses concurrent processes in terms of *agents* and *agent expressions* in a certain language, and defines a single labeled transition system where Σ is the set of *all*, infinitely many (in fact uncountably many) agent expressions, with $\mathcal{T}$ similarly infinite. Strong and weak bisimulation are used to define which agent expressions are in fact "the same agent" in terms of either what actions they can do (strong) or what *observable* (non-τ) actions they can do (weak). The two types of bisimulation are eventually used to define a notion of equality of agent expressions which roughly matches "same sequence of observable actions" while being preserved by each of the combinators in the language used to define an agent expression. For this purpose, the concept of one labeled transition system is particularly useful, and this one labeled transition system has single relations $\sim$, $\approx$, and $=$ defined as "the" strong bisimulation, weak bisimulation, and equality, respectively. In order to get there, however, Milner defines what it means for a relation to be "a" strong or weak bisimulation, then defines "the" strong or weak bisimulation as the largest such relation. For example,

Definition 5.1 *(Definition 5.5 in [19]).* A relation $\leftrightarrow \subset \Sigma \times \Sigma$ is a (weak) bisimulation if $P \leftrightarrow Q$ implies, for all $\alpha \in \mathcal{T}$,

(i) Whenever $P \xrightarrow{\alpha} P'$ then, for some Q', $Q \overset{\alpha}{\Rightarrow} Q'$ and $P' \leftrightarrow Q'$
(ii) Whenever $Q \xrightarrow{\alpha} Q'$ then, for some P', $P \overset{\alpha}{\Rightarrow} P'$ and $P' \leftrightarrow Q'$

where, as in Section 3.1, $P \overset{\tau}{\Rightarrow} Q \iff P \xrightarrow{\tau}{}^{*} Q$ and $P \overset{\alpha}{\Rightarrow} Q \iff P \xrightarrow{\tau}{}^{*} P'' \xrightarrow{\alpha} Q'' \xrightarrow{\tau}{}^{*} Q$ for $\alpha \neq \tau$. (Our notation is slightly different from Milner's: we use $\overset{\alpha}{\Rightarrow}$ to mean the same thing as Milner's $\overset{\hat{\alpha}}{\Rightarrow}$.) Note the similarity between Definitions 5.1 and 3.2(III).

In contrast to Milner in [19] comparing two agents (states) of the same labeled transition system, we want to compare two CRNs which we think of as separate (labeled transition) systems. This itself is not a significant difference: given two systems $(\Sigma_1, \mathcal{T}, \xrightarrow{t}_1 : t \in \mathcal{T})$ and $(\Sigma_2, \mathcal{T}, \xrightarrow{t}_2 : t \in \mathcal{T})$ and a relation $\leftrightarrow \subset \Sigma_1 \times \Sigma_2$ we can consider the system $(\Sigma = \Sigma_1 \cup \Sigma_2, \mathcal{T}, \xrightarrow{t} = \xrightarrow{t}_1 \cup \xrightarrow{t}_2 : t \in \mathcal{T})$ with $\leftrightarrow \subset \Sigma_1 \cup \Sigma_2 \subset \Sigma \times \Sigma$, fitting Milner's paradigm with no significant changes. More importantly, our concept of CRN equivalence wants to consider an asymmetric pair of CRNs: one, $(\mathcal{S}, \mathcal{R})$, is the "formal" CRN where $\mathcal{R}$ is the set of "meaningful actions", and another, $(\mathcal{S}', \mathcal{R}')$, is meant to be an implementation of the formal CRN. This means that some natural conditions we want our definition of correctness to have are that every state of the implementation CRN corresponds to one and only one state of the formal CRN; that every state of the formal CRN has *at least* one state of the implementation CRN that implements it; and since we're working with CRNs which are fundamentally linear, that the sum of any number of implementation states corresponds to the sum of their corresponding formal states. (This linearity condition is also why the undecidability result from [21] doesn't apply to our CRN bisimulation.) It turns out that those conditions on a relation $\leftrightarrow \subset \mathbb{N}^{\mathcal{S}} \times \mathbb{N}^{\mathcal{S}'}$ are true if and only if $\leftrightarrow$ corresponds to some interpretation $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ as defined in Definition 3.1:

Lemma 5.1. *Let $\leftrightarrow \subset \mathbb{N}^{\mathcal{S}} \times \mathbb{N}^{\mathcal{S}'}$ be a relation between formal states and implementation states. If for every implementation state S' there is exactly one formal state S such that $S \leftrightarrow S'$ (function) and for every pair of pairs $S_1 \leftrightarrow S_1'$ and $S_2 \leftrightarrow S_2'$ we have $S_1 + S_2 \leftrightarrow S_1' + S_2'$ (linearity), then there is some interpretation $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ which, when extended to implementation states $m : \mathbb{N}^{\mathcal{S}'} \to \mathbb{N}^{\mathcal{S}}$, induces that relation: $S \leftrightarrow S' \iff S = m(S')$. Furthermore, for every S there is some S' such that $S \leftrightarrow S'$ (surjectivity) iff m satisfies the atomic condition.*

Proof. Given that the relation $\leftrightarrow$ is a linear function from $\mathbb{N}^{\mathcal{S}'}$ to $\mathbb{N}^{\mathcal{S}}$, we define the interpretation to be $m(x) = S_x$ where S_x is the unique formal state such that $S_x \leftrightarrow \{\!|x|\!\}$. Now, any implementation state S' is some sum of implementation species, $S' = \sum_{x \in \mathcal{S}'} \alpha_x x$, and because we define the interpretation of a state as the sum of interpretations of species, $m(S') = \sum_{x \in \mathcal{S}'} \alpha_x m(x)$. Then by the linearity assumption on $\leftrightarrow$, $m(S') \leftrightarrow S'$. Thus, if $S = m(S')$, then $S \leftrightarrow S'$. Conversely, if $S \leftrightarrow S'$, then $S = m(S')$ because $\leftrightarrow$ is a function.

If we further assume that $\leftrightarrow$ is surjective, then in particular for each formal species A, there must be some S' such that $\{\!|A|\!\} \leftrightarrow S'$, i.e. $m(S') = \{\!|A|\!\}$. Since $m(S')$ is the sum of interpretations of species in S' and an implementation species cannot interpret to fractional or negative formal species, there must be some species $x_A \in S'$ with $m(x_A) = \{\!|A|\!\}$ (and any other species in S' interpret to $\emptyset$). Thus the atomic condition is satisfied. Conversely, if the atomic condition is satisfied, then consider an arbitrary formal state $S = \sum_{A \in \mathcal{S}} \alpha_A A$. Using linearity, let $S' = \sum_{A \in \mathcal{S}} \alpha_A x_A$, so $m(S') = S$, and thus $\leftrightarrow$ must be surjective. □

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

Since we said that $\mathcal{R}$ should be the set of "meaningful actions", that means our transition systems $(\Sigma_i, \mathcal{T}, \xrightarrow{t}_i)$ should have the same set of labels, and at first glance that set of labels should be $\mathcal{T} = \mathcal{R} \cup \{\tau\}$. In the formal CRN, this is easy: the interpretation of a CRN as a transition system that we previously described has $\mathcal{T} = \mathcal{R}$, which simply means no τ transitions appear. In the implementation CRN, we need to find a correspondence between implementation reactions and formal reactions (or τ), but this is already what the interpretation does: as described in Definition 3.1, any interpretation $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ induces a map $m : \mathcal{R}' \to (\mathbb{N}^{\mathcal{S}} \times \mathbb{N}^{\mathcal{S}}) \cup \{\tau\}$. (We previously referred to members of $\mathbb{N}^{\mathcal{S}} \times \mathbb{N}^{\mathcal{S}}$ not necessarily in $\mathcal{R}$ as "reactions in the language of the formal CRN".) Since an implementation reaction might be interpreted as a reaction in the language of the formal CRN which is "invalid" i.e. not in $\mathcal{R}$, we take $\mathcal{T} = (\mathbb{N}^{\mathcal{S}} \times \mathbb{N}^{\mathcal{S}}) \cup \{\tau\}$, and when converting the implementation CRN to a transition system we say for $r \in \mathcal{T}$ that $S' \xrightarrow{r} T'$ if $S' \xrightarrow{r'} T'$ and $m(r') = r$ for some $r' \in \mathcal{R}'$. (This means that, formally, which transition systems we are comparing depends on the relation we find between them, a bit of apparent circularity which leads to various differences between CRN bisimulation and the classic definition.) Given this, an interpretation is a CRN bisimulation (satisfies Definition 3.2(III)) if and only if it satisfies the Atomic Condition (Definition 3.2(II.i)) and the relation on states it induces is a weak bisimulation (Definition 5.1). Therefore, a valid interpretation m can be equivalently described as a surjective linear weak bisimulation.

We would like to compare some features of our concept of CRN bisimulation to bisimulation in transition systems as in [19]. To avoid ambiguity, for this discussion we use the phrase "bisimulation relation" to mean a relation between states $\leftrightarrow \subset S \times S$ which satisfies Definition 5.1, and "CRN bisimulation" or "bisimulation interpretation" to mean an interpretation $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ which satisfies Definition 3.2 and therefore induces a bisimulation relation.

The most important difference between a bisimulation relation and a CRN bisimulation is that, as we said earlier, the transition system induced by the implementation CRN is not fully defined until an interpretation is given. For example, if $x_1 \to x_2 \in \mathcal{R}'$, $m(x_1) = A$ and $m(x_2) = B$ then the transition $\{\!|2x_1|\!\} \to \{\!|x_1, x_2|\!\}$ has label $A \to B$, but if $m(x_1) = m(x_2) = A$ then the same transition has label τ. So for example, the fact (Proposition 5.1(4) in [19]) that $\bigcup_{i \in I} \leftrightarrow_i$ is a bisimulation relation if each $\leftrightarrow_i$ is a bisimulation relation has no obvious analog for CRN bisimulations, since there is no obvious way to even define the union of two relations defined on different and "contradictory" transition systems. We don't try.

Milner discusses the relation $\approx = \bigcup\{\leftrightarrow | \leftrightarrow \text{ is a bisimulation relation}\}$ [19]. In any given transition system, such a relation exists, is the largest bisimulation relation (which follows from the previous statement about unions), and is an equivalence relation. In the context of comparing agent expressions, this is a useful way of saying, for example, that the result of applying a sequence of transitions to a complex agent expression is another complex agent expression. In the context of a formal CRN $(\mathcal{S}, \mathcal{R})$ and its induced transition system $(\mathbb{N}^{\mathcal{S}}, \mathcal{R} \cup \{\tau\}, \xrightarrow{r})$, the relation $\approx$ exists but is not very useful. In fact the relation is the trivial $S \approx T \iff S = T$ relation except in some particularly degenerate CRNs; for example, if $\mathcal{S} = \{A\}$ and $\mathcal{R} = \{\emptyset \to A\}$ then $S \approx T$ for all S, T. In the context of CRN bisimulation, a formal CRN and implementation CRN where the actions are reactions in the language of the formal CRN or τ, as previously discussed we haven't finished defining the transition system, so $\approx$ is not yet defined without an interpretation. Given an interpretation m, we have a transition system so $\approx$ exists, but it is mostly restricted by m. If m is a CRN bisimulation, then adopting the convention that $m(S) = S$ for formal states $S \in \mathbb{N}^{\mathcal{S}}$, $S \approx T \iff m(S) \approx m(T)$ for any (formal or implementation) states S, T, where the right side can use the definition of $\approx$ on the formal CRN. That is, "the bisimulation" is just m together with any degeneracy in the formal CRN.

5.2. Handling spurious catalysts

The definition of CRN bisimulation as stated previously has difficulty handling overly detailed enumerations of DNA strand displacement circuits, but a simple extension of bisimulation fixes that problem. As an example, consider the implementation of the reaction $A + B \to C + D$ according to the variant of the scheme by Soloveichik et al. [14] discussed in Section 3. Fig. 15 shows a spurious reaction possible in that scheme: a toehold on the "trigger strand" (what would be t_{CD} if released) in the complex i_A binds to the exposed complementary toehold in another copy of i_A. This spurious binding has no meaningful effect on the DSD system's function: no strand displacement reactions are possible given that binding that should not be possible, and since the binding is reversible it can fall off before any reaction that it would otherwise interfere with. In particular, an analog of the $i_A + x_B \to t_{CD} + w_1$ reaction can still happen in this complex, producing a t_{CD} strand with a spurious binding to an i_A complex (and a normal w_1 waste complex). However, when analyzing the system with bisimulation, we need to interpret each implementation species, including this complex and the result of the reaction. In order for the binding and unbinding reactions to be trivial, we must interpret the $i_A : i_A$ complex as $2A$ and the $t_{CD} : i_A$ complex as $C + D + A$, and they must be trivial in order to satisfy the delimiting condition. Then the reaction $i_A : i_A + x_B \to t_{CD} : i_A + w_1$ is interpreted as $2A + B \to A + C + D$. This is neither trivial nor is it a formal reaction, so by bisimulation as so far defined, the delimiting condition is violated. However, it is "clearly" the reaction $A + B \to C + D$ with a "spurious catalyst" A on the side, and we would like a definition of bisimulation that can confirm this.

Recall that in Definition 3.1 we defined three related but distinct interpretations: $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ an interpretation of implementation *species*; $m : \mathbb{N}^{\mathcal{S}'} \to \mathbb{N}^{\mathcal{S}}$ an interpretation of implementation *states*; and $m : \mathbb{N}^{\mathcal{S}'} \times \mathbb{N}^{\mathcal{S}'} \to (\mathbb{N}^{\mathcal{S}} \times \mathbb{N}^{\mathcal{S}}) \cup \{\tau\}$ an interpretation of implementation *reactions*. At the time, we said that the interpretation of species $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ was arbitrary, but the other two were defined unambiguously in terms of that m. While we still want to keep the interpretation of *states* as the sum of the interpretations of species, we can loosen the definition of the interpretation of a reaction to allow reactions like $i_A : i_A + x_B \to t_{CD} : i_A + w_1$ to be interpreted as $A + B \to C + D$ as intended.

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

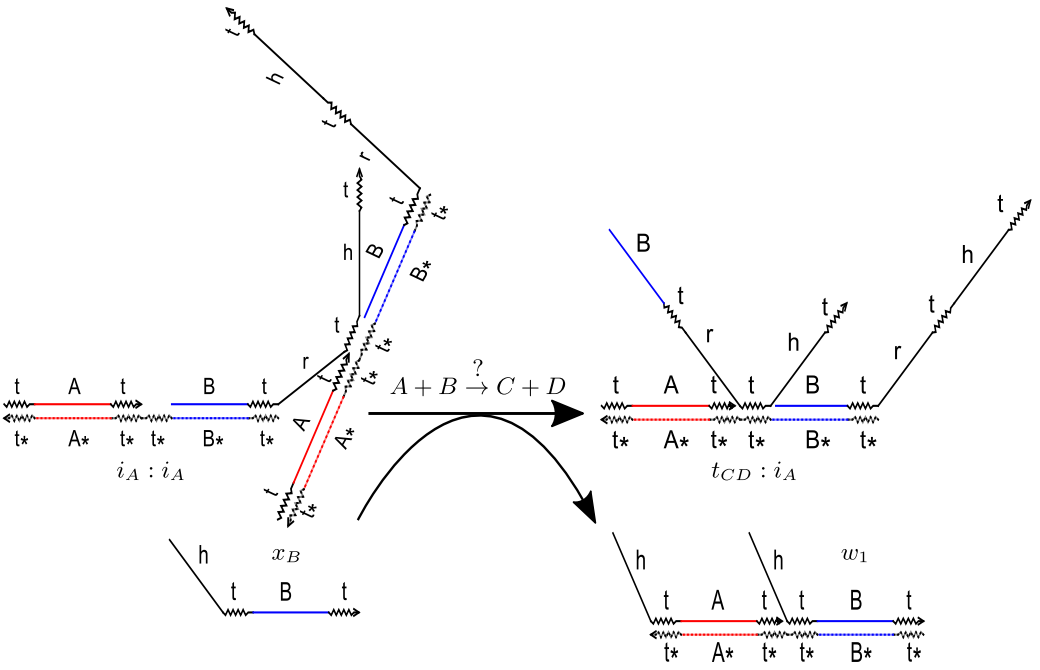


Fig. 15. An example spurious reaction in a variant of the translation scheme by Soloveichik et al. [14].

Definition 5.2. 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 3.1. An interpretation of reactions $m_\rho : \mathbb{N}^{\mathcal{S}'} \times \mathbb{N}^{\mathcal{S}'} \to (\mathbb{N}^{\mathcal{S}} \times \mathbb{N}^{\mathcal{S}}) \cup \{\tau\}$ is *consistent with* m if:

(i) If $R' \neq P'$ but $m(R') = m(P')$ then $m_\rho(R' \to P') = \tau$, and
(ii) If $m(R') \neq m(P')$ and $m_\rho(R' \to P') = R \to P$ then there is some $C \in \mathbb{N}^{\mathcal{S}}$ such that $m(R') = R + C$ and $m(P') = P + C$.

Adapting the definition of bisimulation to this new concept of interpretation is straightforward.

Definition 5.3. Let $(\mathcal{S}, \mathcal{R})$ and $(\mathcal{S}', \mathcal{R}')$ be a formal and implementation CRN. Let $m : \mathcal{S}' \to \mathbb{N}^{\mathcal{S}}$ be an interpretation of implementation species and m_ρ an interpretation of reactions consistent with m. The pair (m, m_ρ) is a CRN bisimulation if m satisfies the atomic condition, m_ρ satisfies the delimiting condition, and the combination satisfies the permissive condition where m is applied to states and m_ρ to reactions.

Most of the important properties of the interpretation of reactions remain true for any m_ρ consistent with m. For example, if $S' \xrightarrow{r'} T'$ and $m_\rho(r') = r = R \to P$ then $m(T') = m(S') - R + P$, as we used in proving Theorem 3.1. In fact, once trajectory equivalence and weak bisimulation are redefined to use m_ρ for the interpretation of any reaction, they remain equivalent to the three conditions by the same logic used in that theorem. Less trivially, it turns out that the algorithms for checking or finding a bisimulation discussed in Section 4 can be modified to work with the new concept of CRN bisimulation. (For complexity purposes, we assume m_ρ is written by writing the index of $m_\rho(r')$ for each $r' \in \mathcal{R}'$, giving it size $|m_\rho| \leq |\mathcal{R}'| \log |\mathcal{R}| \leq n \log n$.)

Theorem 5.1. *Given a formal and implementation CRN $(\mathcal{S}, \mathcal{R})$ and $(\mathcal{S}', \mathcal{R}')$ with interpretation (m, m_ρ) where m_ρ is consistent with m, the problem of checking whether (m, m_ρ) is a CRN bisimulation is PSPACE-complete in general, and can be checked in polynomial space by the* loopsearch *algorithm as previously described. The* graphsearch *algorithm as previously described also correctly checks whether (m, m_ρ) is a CRN bisimulation, and when the number of reactants in any formal reaction in $\mathcal{R}$ is bounded by some k, the* graphsearch *algorithm runs in* poly(n^k) *time and space.*

Proof. Both algorithms do not depend on the fact that $m(R' \to P') = m(R') \to m(P')$ or τ, but only depend on being able to find $m(r')$ given r', which is still easy given m_ρ. Similarly, assumptions made by the algorithm such as if $S' \xrightarrow{\tau} T'$ then $m(S') = m(T')$ still hold. Thus, the previous proof of correctness and complexity of the algorithms holds. Similarly, in the completeness proof in Theorem 4.3, the given interpretation with $m_\rho(R' \to P') = m(R') \to m(P')$ when $m(R') \neq m(P')$ is

consistent with m and is the same as the interpretation in that theorem, and is still correct if and only if the space-bounded Turing machine accepts, so the problem is still PSPACE-complete. □

When we try to find an interpretation, we modify the reactionsearch algorithm as follows. The algorithm takes as input $(\mathcal{S},\mathcal{R})$ and $(\mathcal{S}',\mathcal{R}')$ as the previous version, and also takes zero or more *interpretation constraints*. An interpretation constraint is a statement of one of the forms $m(x)=S_x$, $m(x)\geq S_x$, or $m_\rho(r')=r$, where x is an implementation species, S_x a multiset of formal species, r' an implementation reaction, and r a formal reaction. We assume any r' has zero or one $m_\rho(r')=r$ constraints and any x has either exactly one $m(x)=S_x$ constraint or zero or more $m(x)\geq S_x$ constraints but not both; anything else would be redundant or contradictory, and it would be easy to tell which it is. Where the algorithm previously would consider the possibility that $m(r')=r$ and enumerate all possible partial interpretations of uninterpreted species in r', then call itself recursively, the new algorithm enumerates all possible *minimal* (in the usual sense of having no strict subset be valid; e.g. if $m(x_1)\geq A$, $m(x_2)\geq B$ is valid then $m(x_1)\geq 2A$, $m(x_2)\geq B$ is not minimal) partial interpretations of all implementation species x appearing in r' which do not have an $m(x)=S_x$ constraint, and recursively calls itself with the reaction constraint $m_\rho(r')=r$ and the enumerated partial interpretation encoded as $m(x)\geq S_x$ constraints, with an exception: if every species x on one side of r' has an $m(x)=S_x$ constraint, then the enumerated interpretation of the other side is passed as $m(x)=S_x$ constraints.

The algorithm runs the trivial reaction solver when every r' with no $m_\rho(r')=r$ constraint can be trivial. When solving the atomic condition before running the trivial reaction solver, the new algorithm assigns an $m(x)=A$ for each A such that there is no $m(x)=A$ restriction already present, and chooses from all x such that there is no $m(x)=S_x\neq A$ restriction and $m(x)=A$ is not contradicted by any $m(x)\geq S_x$ restriction. When running the trivial reaction solver, the algorithm first assigns to each x with no $m(x)=S_x$ restriction a "base interpretation" $m_0(x)=\bigvee_{m(x)\geq S_x} S_x$, then sets up and solves equations in terms of an "additional interpretation" $m_+(x)$ such that $m(x)=m_0(x)+m_+(x)$. (As in the previous algorithm, it solves separately for each $m_+(x;A)$.) The algorithm sets up an equation for *every* implementation reaction r', even those with an $m_\rho(r')=r\neq\tau$ constraint; if $r'=R'\to P'$ has an $m_\rho(r')=r=R\to P$, then the equations are of the form $m(R')-R-m(P')+P=0$, expanded in terms of each $m_+(x;A)$ after subtracting m_0 and replacing any species x with an $m(x)=S_x$ constraint. As before, the algorithm searches for only one minimal solution, then tests the permissive condition, since as before if there is any solution that satisfies the permissive condition then there is a unique (for every formal species that appears as a reactant in any reaction) minimal solution and it satisfies the permissive condition.

Theorem 5.2. *Given a formal and implementation CRN $(\mathcal{S},\mathcal{R})$ and $(\mathcal{S}',\mathcal{R}')$ with zero or more conditions of the form $m(x)=S_x$, $m(x)\geq S_x$, or $m_\rho(r')=r$, the modified reactionsearch algorithm as described above correctly finds an interpretation or asserts that none exists. The algorithm runs in polynomial space, and if a correct interpretation exists then the algorithm outputs one that is polynomial size in its inputs. Deciding whether an interpretation from a given implementation CRN to a given formal CRN is PSPACE-complete in the general case, and is NP-complete when the number of reactants in any reaction in $\mathcal{R}$ is bounded by a constant $k\geq 1$.*

Proof. Lemma 4.4 still applies to the new trivial reaction solver; the proof as stated applies exactly to the new definition of CRN bisimulation with the exception that the statement $m(Y')(A)=R(A)>m(R')(A)$ needs to become $m(Y')(A)\geq R(A)>m(R')(A)$, which does not affect the remainder of the proof. The proof that the new algorithm is correct then follows the same lines as the proof that the old algorithm is correct in Theorem 4.4; if a correct completion of the interpretation exists, then one of the branches of the algorithm will find it.

To prove that finding an interpretation is PSPACE-complete, we use the same reduction from linear bounded Turing machine acceptance to formal and implementation CRN as in Theorem 4.5. Even under the expanded definition of correct interpretation, the interpretation $m(0_i)=m(1_i)=A_i$, $m(h)=H$, $m(q_i^j)=Q$ for $j\neq m-1$ and $m(q_i^{m-1})=Q+\sum_{k<i}A_k$, with $m_\rho(q_n^{m-1}+0_n\to h)=Q+A_1+\cdots+A_n\to H$ and $m_\rho(r')=\tau$ otherwise, is the only possibly correct interpretation up to a permutation of formal species Q and the A_i's, and it is correct if and only if the given Turing machine accepts the given input string x. The proof, however, requires some different steps to rule out possibilities that are opened up by a less restrictive definition of interpretation. First, we observe that $m_\rho(q_i^m+x_i\to q_{i-1}^m+x_i)=\tau$, since combining those reactions with $q_i^m\to q_n^m$ gives a loop from $q_n^m+x_1+\cdots+x_n$ to itself; this is impossible in the formal CRN if any reaction fires, so all reactions involved must be trivial (or violate the delimiting condition). This means that $m(q_i^m)=m(q_n^m)$, and applying the same to $q_1^m+x_1\to q_1^0+x_1$ and $q_1^0\to q_n^m$ we get that $m(q_1^0)=m(q_n^m)$ also. Now, each $q_i^m+x_{-i}\to q_{i-1}^m+x_i$ and $q_1^m+x_{-1}\to q_1^0+x_1$ reaction shows that for each i either $m(0_i)=m(1_i)$ or $m(x_i)=H$ and $m(x_{-i})=Q+A_1+\cdots+A_n$ (the second case was impossible in the old definition when $m(q_i^m)\neq\emptyset$), and similarly from $q_i^j\to q_n^m$ either $m(q_i^j)=Q+A_1+\cdots+A_n$ or $m(q_i^j)=m(q_n^m)$. Since no formal reaction is reversible, we have $m_\rho(q_1^{m-1}+1_1\rightleftharpoons q_2^{m-1})=m_\rho(q_{i-1}^{m-1}+0_{i-1}\rightleftharpoons q_i^{m-1})=\tau$, so $m(q_i^{m-1})=m(q_1^{m-1})+m(1_i)+\sum_{2\leq k<i}m(0_k)$ for $i\geq 2$. This is important because, as in the previous proof, we have $n+2$ formal species and by the atomic condition, each one must have at least one implementation species interpreted as one copy of it and nothing else. Although we have yet to prove that $m(0_i)=m(1_i)$, the above does prove that the two cannot (for the same i) satisfy the atomic condition for two *different* formal species; similarly, no q_i^j can satisfy the atomic condition for a different formal species than q_n^m. This leaves $n+2$ groups of implementation species—q_n^m, x_i for $1\leq i\leq n$, and h—to $n+2$ formal species, so each of the mentioned implementation species must be interpreted as exactly one copy

of a different formal species. Then $m(0_n)$ is either a single formal species not equal to $m(h)$, or is $Q + A_1 + \cdots + A_n$, so $m_\rho(q_n^{m-1} + 0_n \to h) \neq \tau$, so $m_\rho(q_n^{m-1} + 0_n \to h) = Q + A_1 + \cdots + A_n \to H$ and $m(h) = H$. This rules out the possibilities that any of the x_i's or q_n^m are interpreted as H, leaving the only possibility that they are interpreted as some assignment of Q and the A_i's, with $m(0_i) = m(1_i)$, $m(q_i^j) = m(q_n^m)$ for $j \neq m-1$, and $m(q_i^{m-1}) = m(q_n^m) + \sum_{k<i} m(0_k)$. This interpretation satisfies the atomic and delimiting conditions, and satisfies the permissive condition if and only if the given Turing machine accepts x, thus deciding whether a correct interpretation exists is still PSPACE-complete.

The proof of Theorem 4.6 applies to the new definition of interpretation without modification, proving that whether a correct interpretation exists is NP-complete when the number of reactants in a formal reaction is bounded by a constant $k \geq 1$. □

6. Discussion

Comparing Chemical Reaction Networks on different levels of abstraction is an important tool for systematic programming with CRNs. We showed how to adapt the concept of bisimulation to check whether one CRN is a correct implementation of another. We showed that bisimulation can be used to prove the correctness of some existing CRN implementations, and to identify subtle but real problems with others. We discussed transitivity and modularity, which can be used to simplify a bisimulation proof. We presented different algorithms to check bisimulation which are adapted to different cases. We showed that the condition can be checked in polynomial time with favorable assumptions, is NP-complete with less favorable assumptions, and is PSPACE-complete in the general case.

In the beginning, we mentioned a DNA implementation of the approximate majority CRN [7] that was experimentally demonstrated by Chen et al. [3]. We might consider applying our bisimulation checker to this implementation. The implementation as presented in [3] would be incorrect according to bisimulation, for the same reason the example in Fig. 5 from Qian et al. [15] fails: outputs of an irreversible reaction are released before an irreversible step is taken, leading to a small probability of such a reaction reversing itself after the products have reacted downstream. Despite this, Chen et al.'s *in vitro* experimental demonstration showed no such problems. While there are a number of explanations for this observation, including that the formal approximate majority CRN is particularly resistant to error [7], it nonetheless raises the question of how serious are the potential errors that may occur in CRN implementations that are not correct according to bisimulation? The answer will depend on the specific formal CRN of interest, as well as the conditions under which it is run. For example, behavior that may be problematic with non-negligible probability in low molecular counts, may have negligible effect in high molecular counts typical of *in vitro* experiments.

Another observation we have is that for typical engineered CRN implementations, at least for DNA strand displacement implementations, either there is a problem in the implementation of one formal reaction; or there is a problem with crosstalk between formal reactions; or there is no problem, and correctness can be proven by the modularity condition. In the case of crosstalk, as we mentioned in Section 3.4, that problem needs to be detected by the reaction enumerator, and is beyond the scope of our bisimulation theory. In the implementation by Chen et al. [3], for example, there are three formal reactions, but the (technically) incorrect behavior can be detected by considering only one of them. In the implementation of the rock–paper–scissors oscillator by Srinivas et al. [4], they use a systematic translation method slightly modified from Soloveichik et al. [14]. After confirming that their method applied to one reaction is correct, using Corollary 3.3 we can prove that such a method applied to *any* combination of reactions will be correct according to bisimulation.

The theory and algorithms discussed in this paper have been incorporated by Badelt et al. into the Nuskell compiler, a software package that automatically translates a CRN into a DNA strand displacement circuit and verifies that the result is correct [42]. Nuskell currently contains the loopsearch and graphsearch algorithms for checking the permissive condition as well as an exhaustive search algorithm for the same, the reactionsearch algorithm for finding an interpretation, and the algorithms to check the modularity condition and find a modular interpretation when given a decomposition into modules of an implementation CRN. Badelt et al. use bisimulation to verify a number of translation schemes applied to the rock–paper–scissors oscillator [5,6,4], showing that bisimulation algorithms can be used to verify CRN implementations used in practice.

Algorithms such as the graphsearch algorithm and loopsearch algorithm scale better with the number of meaningful species than the number of null species, while engineered CRN implementations generally do not use loops that produce null species. Thus those algorithms will be faster than their worst-case limits in practical cases. For example, the graphsearch algorithm takes at most $(2zn^k + 1)n^k = O(n^{2k+1})$ cycles in theory, where n is the number of implementation species, k the largest number of reactants in a formal reaction, and z the number of implementation species with empty interpretation. When there are no null species (or when none can be produced in a loop, as in schemes such as [14]), this becomes at most n^k cycles.

In CRN bisimulation, we require that *every* implementation species has an interpretation as a (possibly empty) multiset of formal species. In contrast, verification methods such as pathway decomposition [17] or serializability [18] both assume that each formal species is represented by one implementation species, while other implementation species are classified into fuels, wastes, and intermediates. Because of this, pathway decomposition and serializability compare formal reactions to implementation pathways which begin and end with (representations of) formal species, while in bisimulation an individual implementation reaction can be interpreted and compared to the formal CRN. An additional consequence, for pathway decomposition, is that correctness guarantees do not apply to implementation states that cannot be reached from initial

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002

states representing formal species, whereas bisimulation is more robust in that correctness is asserted in those cases as well. Furthermore, even in the permissive condition, bisimulation requires that there *exist* an implementation pathway which implements a given formal reaction, while pathway decomposition and serializability both require that *all* implementation pathways have properties which may be nontrivial to check. This locality is what allows us to prove the complexity results given, which we suspect are significantly lower complexity than methods that depend on implementation pathways.

However, the use of interpretations instead of pathways means that in some cases CRN bisimulation and pathway decomposition differ on which implementations they consider correct. Bisimulation can easily be adapted to situations where there is no clear single "canonical representation" of a given formal species, while pathway decomposition has difficulty. For example, the implementation in [15] of the *reversible* formal reaction $A + B \rightleftharpoons C + D$ by reversible implementation reactions $\{x_A \rightleftharpoons i_A, i_A + x_B \rightleftharpoons i_{CD}, i_{CD} \rightleftharpoons x_C + i_D, i_D \rightleftharpoons x_D\}$. Bisimulation considers this correct with the obvious interpretation, while pathway decomposition considers the ability to release x_C then reverse without releasing x_D to be an error. On the other hand, bisimulation has trouble handling implementation species with no well-defined interpretation. Shin et al. describe a "delayed choice" phenomenon where an implementation CRN commits to implementing one of two formal reactions before deciding which one, producing an intermediate that cannot be correctly interpreted as either of the reaction's products or their reactants; such implementations are generally considered incorrect according to bisimulation but pathway decomposition often considers them correct [17]. They then propose a hybrid notion of correctness where an implementation CRN is considered correct if it is a correct implementation according to pathway decomposition of some intermediate CRN, and the intermediate CRN is a correct implementation of the formal CRN according to bisimulation [17]. This notion considers correct any implementation that is correct according to either pathway decomposition or bisimulation, plus some others.

One area this theory overlooks is the rates of reactions and the probabilities of reaching certain states. For example, in [14] Soloveichik et al. argue that the concentration of each intermediate is proportional to the product of that of the formal species which we would call its interpretation, and thus the reaction rates are approximately correct. Whether this can be generalized, and whether bisimulation can help this generalization, is an important open question.

Acknowledgements

The authors would like to thank Chris Thachuk, Damien Woods, Dave Doty, and Seung Woo Shin for helpful discussions. We would also like to thank the anonymous reviewers for many helpful suggestions. RFJ and EW were supported by NSF grants 1317694, 1213127, and 0832824. RFJ was supported by Caltech's Summer Undergraduate Research Fellowship program and an NSF graduate fellowship. QD thanks Steve Skiena for his kindness and flexibility.

References

[1] L. Cardelli, Morphisms of reaction networks that couple structure to function, BMC Syst. Biol. 8 (2014) 1–18.
[2] L. Cardelli, A. Csikász-Nagy, The cell cycle switch computes approximate majority, Sci. Rep. 2 (2012), https://doi.org/10.1038/srep00656.
[3] 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.
[4] N. Srinivas, J. Parkin, G. Seelig, E. Winfree, D. Soloveichik, Enzyme-free nucleic acid dynamical systems, Science 358 (2017), https://doi.org/10.1126/science.aal2052.
[5] 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: Proceedings of the Third European Conference on Artificial Life, Granada, Spain, June 4–6, 1995, Springer, 1995, pp. 784–800.
[6] A. Dobrinevski, E. Frey, Extinction in neutrally stable stochastic Lotka–Volterra models, Phys. Rev. E 85 (2012) 051903.
[7] D. Angluin, J. Aspnes, D. Eisenstat, A simple population protocol for fast robust approximate majority, Distrib. Comput. 21 (2008) 87–102.
[8] H.-L. Chen, D. Doty, D. Soloveichik, Deterministic function computation with chemical reaction networks, Nat. Comput. 13 (2014) 517–534.
[9] D. Soloveichik, M. Cook, E. Winfree, J. Bruck, Computation with finite stochastic chemical reaction networks, Nat. Comput. 7 (2008) 615–633.
[10] D.Y. Zhang, G. Seelig, Dynamic DNA nanotechnology using strand-displacement reactions, Nat. Chem. 3 (2011) 103–113.
[11] H.-L. Chen, D. Doty, D. Soloveichik, Rate-independent computation in continuous chemical reaction networks, in: M. Naor (Ed.), Proceedings of the 5th Conference on Innovations in Theoretical Computer Science, ACM, 2014, pp. 313–326.
[12] 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.
[13] D. Doty, M. Hajiaghayi, Leaderless deterministic chemical reaction networks, Nat. Comput. 14 (2015) 213–223.
[14] D. Soloveichik, G. Seelig, E. Winfree, DNA as a universal substrate for chemical kinetics, Proc. Natl. Acad. Sci. 107 (2010) 5393–5398.
[15] 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.
[16] L. Cardelli, Two-domain DNA strand displacement, Math. Structures Comput. Sci. 23 (2013) 247–271.
[17] S.W. Shin, C. Thachuk, E. Winfree, Verifying chemical reaction network implementations: a pathway decomposition approach, Theoret. Comput. Sci. (2017), https://doi.org/10.1016/j.tcs.2017.10.011.
[18] M.R. Lakin, D. Stefanovic, A. Phillips, Modular verification of chemical reaction network encodings via serializability analysis, Theoret. Comput. Sci. 632 (2016) 21–42.
[19] R. Milner, Communication and Concurrency, Prentice-Hall, Inc., 1989.
[20] J.-C. Fernandez, An implementation of an efficient algorithm for bisimulation equivalence, Sci. Comput. Program. 13 (1990) 219–236.
[21] P. Jančar, Undecidability of bisimilarity for Petri nets and some related problems, Theoret. Comput. Sci. 148 (1995) 281–301.
[22] M. Antoniotti, C. Piazza, A. Policriti, M. Simeoni, B. Mishra, Taming the complexity of biochemical models through bisimulation and collapsing: theory and practice, Theoret. Comput. Sci. 325 (2004) 45–67.
[23] 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.

[24] C. Grun, K. Sarma, B. Wolfe, S.W. Shin, E. Winfree, A domain-level DNA strand displacement reaction enumerator allowing arbitrary non-pseudoknotted secondary structures, CoRR, arXiv:1505.03738, 2015.
[25] S. Gay, S. Soliman, F. Fages, A graphical method for reducing and relating models in systems biology, Bioinformatics 26 (2010) i575, https://doi.org/10.1093/bioinformatics/btq388.
[26] L. Cardelli, M. Tribastone, M. Tschaikowski, A. Vandin, Forward and backward bisimulations for chemical reaction networks, in: L. Aceto, D. de Frutos Escrig (Eds.), 26th International Conference on Concurrency Theory, CONCUR 2015, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 42, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2015, pp. 226–239.
[27] L. Cardelli, M. Tribastone, M. Tschaikowski, A. Vandin, Comparing chemical reaction networks: a categorical and algorithmic perspective, in: Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, ACM, 2016, pp. 485–494.
[28] L. Cardelli, M. Tribastone, M. Tschaikowski, A. Vandin, Syntactic Markovian bisimulation for chemical reaction networks, in: L. Aceto, G. Bacci, G. Bacci, A. Ingólfsdóttir, A. Legay, R. Mardare (Eds.), Models, Algorithms, Logics and Tools: Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday, Springer International Publishing, Cham, 2017, pp. 466–483.
[29] L. Cardelli, M. Tribastone, M. Tschaikowski, A. Vandin, Efficient syntax-driven lumping of differential equations, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2016, pp. 93–111.
[30] L. Cardelli, M. Tribastone, M. Tschaikowski, A. Vandin, ERODE: a tool for the evaluation and reduction of ordinary differential equations, in: TACAS (2), 2017, pp. 310–328.
[31] S. Tognazzi, M. Tribastone, M. Tschaikowski, A. Vandin, EGAC: a genetic algorithm to compare chemical reaction networks, in: Proceedings of 1st Genetic and Evolutionary Computation Conference-17, GECCO'17, 2017, pp. 833–840.
[32] R.F. Johnson, Q. Dong, E. Winfree, Verifying chemical reaction network implementations: a bisimulation approach, in: D. Woods, Y. Rondelez (Eds.), DNA Computing and Molecular Programming, in: Lecture Notes in Computer Science, vol. 9818, Springer, 2016, pp. 114–134.
[33] Q. Dong, A Bisimulation Approach to Verification of Molecular Implementations of Formal Chemical Reaction Networks, Master's thesis, Stony Brook University, 2012.
[34] C. Rackoff, The covering and boundedness problems for vector addition systems, Theoret. Comput. Sci. 6 (1978) 223–231.
[35] R. Lipton, The Reachability Problem Requires Exponential Space, Research Report 63, Department of Computer Science, Yale University, New Haven, Connecticut, 1976, pp. 1–15.
[36] W.J. Savitch, Relationships between nondeterministic and deterministic tape complexities, J. Comput. System Sci. 4 (1970) 177–192.
[37] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., New York, NY, USA, 1979.
[38] N.D. Jones, L.H. Landweber, Y.E. Lien, Complexity of some problems in Petri nets, Theoret. Comput. Sci. 4 (1977) 277–299.
[39] C. Thachuk, A. Condon, Space and energy efficient computation with DNA strand displacement systems, in: D. Stefanovic, A. Turberfield (Eds.), DNA Computing and Molecular Programming, in: Lecture Notes in Computer Science, vol. 7433, Springer, 2012, pp. 135–149.
[40] E. Contejean, H. Devie, An efficient incremental algorithm for solving systems of linear Diophantine equations, Inform. and Comput. 113 (1994) 143–172.
[41] L. Pottier, Minimal solutions of linear Diophantine systems: bounds and algorithms, in: International Conference on Rewriting Techniques and Applications, Springer, 1991, pp. 162–173.
[42] S. Badelt, S.W. Shin, R.F. Johnson, Q. Dong, C. Thachuk, E. Winfree, A general-purpose CRN-to-DSD compiler with formal verification, optimization, and simulation capabilities, in: D. Woods, Y. Rondelez (Eds.), DNA Computing and Molecular Programming, in: Lecture Notes in Computer Science, vol. 9818, Springer, 2017, pp. 232–248.

Please cite this article in press as: R. Johnson et al., Verifying chemical reaction network implementations: A bisimulation approach, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.01.002